# Skalierbare und zuverlässige Gruppenkommunikation im Internet

Von der Fakultät Informatik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr.rer.nat.) genehmigte Abhandlung

Vorgelegt von

Christian Maihöfer

aus Mutlangen

| | |
|---|---|
| Hauptberichter: | Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel |
| Mitberichter: | Prof. Dr. rer. nat. habil. Alexander Schill |

Tag der mündlichen Prüfung: 4. Dezember 2002

Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR)
der Universität Stuttgart

2002

# Danksagung

Mein besonderer Dank gilt meinem Doktorvater, Prof. Dr. Kurt Rothermel, der mir die vorliegende Arbeit ermöglicht hat. Sein kritischer wissenschaftlicher Blick als auch die Gewährung großer Freiheiten waren außerordentlich hilfreich und haben wesentlich zum Gelingen dieser Arbeit beigetragen.

Ebenso möchte ich meinem Zweitbetreuer danken, Prof. Dr. Alexander Schill, für die rasche, unkomplizierte und hilfreiche Unterstützung.

Großen Dank schulde ich zudem meinen ehemaligen Kollegen und Freunden in der Abteilung Verteilte Systeme für die hilfreiche und gute Zusammenarbeit, vor allem beim Korrekturlesen von Veröffentlichungen und der vorliegenden Arbeit. Insbesondere hervorzuheben ist hier Alexander Leonhardi, der durch unzählige Diskussionen und vielen wertvollen Kommentaren zum Gelingen meiner Arbeit beigetragen hat.

Schließlich danke ich Isabel Mack für das Korrekturlesen der vorliegenden Arbeit.

Brainkofen, den 20. Dezember 2002

Christian Maihöfer

# Kurzfassung

Multicast erlaubt die bandbreitenschonende Auslieferung einer Nachricht an eine Gruppe von Empfängern. Anwendungen wie Informations– und Softwareverteilung an große Empfängergruppen, Audio– und Videokonferenzen, verteiltes Rechnen und verteilte Spiele lassen sich mit einem Multicast-Dienst effizient realisieren. Eine Reihe von Anwendungen erfordert dabei eine zuverlässige Nachrichtenauslieferung, die über den im Internet verfügbaren unzuverlässigen IP-Multicast-Dienst hinausgeht. Aufbauend auf IP-Multicast wird eine zuverlässige Nachrichtenauslieferung mittels Empfangsbestätigungen und Übertragungswiederholungen durch nicht-hierarchische oder hierarchische Transportprotokolle erreicht.

Wie sich im Verlauf der Arbeit zeigen wird, erlauben lediglich hierarchische Transportprotokolle die *skalierbare* Realisierung eines zuverlässigen Multicast-Diensts zur Gruppenkommunikation. Diese ordnen alle Gruppenmitglieder in eine Hierarchie ein, die Kontrollbaum genannt wird. Durch den Kontrollbaum können notwendige Übertragungswiederholungen lokal begrenzt werden und müssen nicht mehr notwendigerweise vom ursprünglichen Sender durchgeführt werden. Die Verteilung dieser Last auf andere Gruppenmitglieder ermöglicht einen mit der Mitgliederzahl skalierbaren Dienst. Die Voraussetzung für den praktischen Einsatz ist ein skalierbares Verfahren zum Aufbau der Kontrollbäume.

Mit dem Token-Repository-Service (TRS) wird in dieser Arbeit ein effizientes Verfahren zum Aufbau der Kontrollbäume vorgestellt. Drei Realisierungsvarianten des TRS-Diensts, TRS-R, TRS-K und TRS-M, erlauben unterschiedlichste Anforderungen zu berücksichtigen. Das wesentliche Entwurfskriterium aller Varianten ist eine skalierbare und effiziente Realisierung. Vergleichende Berechnungen und Simulationen mit alternativen Ansätzen unterstreichen nicht nur die Tragfähigkeit des TRS-Diensts. Dieser stellt momentan die einzige skalierbare Lösung zum Aufbau von Kontrollbäumen dar. Aufgrund der Unabhängigkeit des TRS-Diensts von einem Multicast-Routing-Protokoll ist es zudem die einzige Lösung, die durchgängig im gesamten Internet eingesetzt werden kann.

Zur Erleichterung der Auswahl eines geeigneten zuverlässigen Multicast-Transportprotokolls und zur optimalen Konfiguration eines ausgewählten Transportprotokolls mit dem TRS-Dienst wird eine ausführliche probabilistische Analyse durchgeführt. Ausgangspunkt der Analyse ist nicht die Untersuchung konkreter Protokolle, sondern die Identifizierung wichtiger generischer Protokollklassen und deren anschließende Beurteilung. Dabei ergeben sich bedeutende Unterschiede bezogen auf den Bandbreitenbedarf, den Durchsatz und die Nachrichtenverzögerungen. Die Evaluation zeigt vielfache Vorteile für hierarchische Verfahren. Diese bieten nicht nur Skalierbarkeit, auch bezüglich des verursachten Nachrichtenaufwands, des Durchsatzes und der erzielbaren Nachrichtenverzögerung sind sie durch eine Verteilung der Last und kürzere Kommunikationswege gegenüber nicht-hierarchischen Protokollen im Vorteil.

Als ein wesentlicher Parameter zur Konfiguration hierarchischer Transportprotokolle wird der Verzweigungsgrad des Kontrollbaums identifiziert werden. Im Gegensatz zu bisherigen Verfahren ermöglicht es der TRS-Dienst den Verzweigungsgrad flexibel zu konfigurieren. Die Analysen und Simulationen zeigen, dass bei einer geeigneten Wahl des Verzweigungsgrads der Durchsatz und die Verzögerung des zuverlässigen Transportprotokolls beträchtlich verbessert werden kann.

Zusammenfassend erlaubt es der vorgestellte Token-Repository-Service durch seinen skalierbaren Kontrollbaumaufbau erstmals, zuverlässige Multicast-Kommunikation auch für große Gruppen zu etablieren. Durch die flexible Wahl des Verzweigungsgrads der entstehenden Kontrollbäume ist zudem eine beträchtliche Leistungsoptimierung des zuverlässigen Transportprotokolls sowie eine entscheidende Verringerung der Netzbelastung mit dem TRS-Dienst möglich.

# Scalable and Reliable Group Communication in the Internet

## English abstract

Multicast allows the bandwidth efficient transmission of messages to a group of receivers. Applications like information and software distribution to large receiver groups, audio and video conferences, distributed computing and network games profit from the low bandwidth consumption of a multicast service. A number of these applications require a reliable multicast service, which is not provided by the best effort IP multicast service currently available in the Internet. A reliable multicast service based on IP multicast introduces acknowledgements und retransmissions. Reliable multicast protocols can be flat or hierarchical approaches.

This thesis shows that only hierarchical transport protocols allow the *scalable* realization of a reliable multicast service. Hierarchical transport protocols organize all group members into a hierarchy, called control tree. The control tree allows that necessary retransmissions are done locally and releases the initial sender from performing all retransmissions. The distribution of the retransmission load between all group members guarantees that the multicast services scales with the number of group members. However, the prerequisite for using hierarchical multicast transport protocols is an efficient and scalable approach for establishing the control trees.

With the Token Repository Service (TRS) an efficient approach for establishing control trees is presented in this thesis. Three realization schemes of the TRS service, TRS-R, TRS-K and TRS-M allow to consider various requirements. The most important design criterion of all schemes is a scalable and efficient approach. Comparative analysis and simulation results show that the TRS service is the only service that scales for large receiver groups and a large number of multicast groups. Moreover, as the TRS service is independent of the underlying multicast routing protocol, it is currently the only service that can be used throughout the Internet.

## Background and Classification of Reliable Multicast Transport Protocols

The following classification is helpful to characterize reliable multicast protocols. The class of sender-initiated protocols is characterized by positive acknowledgments (ACKs) returned by the receivers to the sender. A missing ACK indicates either a lost data packet at the corresponding receiver, a lost ACK packet or a crashed receiver, which cannot be distinguished by the sender. Therefore, a missing ACK packet leads to a data packet retransmission from the sender.

In contrast to sender-initiated protocols, receiver-initiated protocols return only negative acknowledgments (NACKs) instead of ACKs. When a receiver detects an error, e.g. through a wrong checksum, a skip in the sequence number or a timeout while waiting for a data packet, a NACK is returned to the sender. A retransmission can be sent with multicast to the whole group or with unicast to those receivers from which a NACK was received.

Besides sending unicast NACKs, an alternative for receiver-initiated protocols is to send a multicast NACK to the whole group, which is known as NACK-avoidance or NACK-suppression scheme. A receiver that has detected an error waits a random time and then sends a multicast NACK provided that it has not already received a NACK for this data packet from another receiver. Thus, in optimum case, only one NACK is received by the sender for each lost data packet.

While sender-initiated approaches are known to be susceptible to the acknowledgment implosion problem, which means that the sender is overwhelmed by ACK processing, receiver-initiated protocols can mitigate this problem. However, there is still a scalability problem for large multicast groups, since even receiver-initiated protocols may fail in presence of a high message error rate, resulting in a large number of NACKs or due to the crying baby problem. The crying-baby problem is the dependency of receiver-initiated protocols on the most unreliable receiver or most unreliable path to a receiver, respectively. If one receiver permanently detects message loss and sends NACKs, in particular if multicast NACKs are used, the sender and possibly all of this group’s members are congested.

Tree-based approaches promise to overcome the acknowledgment implosion problem and to provide unlimited scalability. They organize the receivers into a tree structure called control tree, which exists for each reliable multicast group. The branching factor is the maximum number of child nodes of a node in the control tree, which limits the load of each node. The control tree is responsible for collecting acknowledgments and sending retransmissions. While the initial multicast messages are sent directly to the receivers by using IP multicast, the responded ACK and NACK messages are aggregated and propagated along the edges of the control tree. For simplicity we assume that

the sender is the root of the control tree. If a receiver needs a retransmission, the parent node in the control tree is informed and asked for a retransmission rather than the sender, which overcomes the acknowledgment implosion problem.

## Related Work

Analysis and simulation studies in this thesis give evidence for the superiority of tree-based approaches compared to non-hierarchical ones. However, the problem remains to create and maintain the control tree. When a new member wants to join the multicast group, it has to connect to the group's control tree. Several techniques have been proposed in the literature for creating the control tree. Most of them are based on or are variants of the expanding ring search (ERS) approach [Boggs 1983], with its main advantages simplicity and fault tolerance.

When a node wants to join the control tree, basic ERS simply multicasts a join request to the members of this group [Yavatkar et al. 1995]. In order to decrease network load and to find a parent in the control tree as close as possible to the searching node, the search is limited by a search scope. The initial request is sent with a time-to-live (TTL) of 1 and thus is limited to the sender's LAN. When a non-occupied node (i.e. a node that can accept further child nodes) in the group's control tree receives this message, it returns an unicast answer. If no node answers within a certain time, the requestor multicasts the join request with an increased TTL. It repeats this process until an answer arrives or the maximum TTL of 255 is reached. The node that answers first becomes the new member's parent in the control tree. TMTP is an example for a tree-based reliable multicast protocol using the ERS approach for tree creation.

Expanding ring advertisement (ERA) is similar to the ERS approach but changes the roles of searching node and searched node. Nodes that are already in the control tree search for children, i.e. new nodes willing to join the control tree [Paul et al. 1997, Chiu et al. 1998]. Non-occupied control tree nodes send multicast invitation messages to the multicast group address. RMTP is an example for a reliable multicast protocol that uses the ERA approach. Some protocols like Lorax or TRAM use a combination of ERS and ERA [Chiu et al. 1998, Levine et al. 1996].

We show in this thesis that ERS and ERA suffer from serious problems. The major problem of both is their limited scalability with large receiver groups. Other problems are that they work only with certain multicast routing protocols, which allow multiple multicast senders, and that they lead to a vast overhead at the routing layer. In this thesis a novel approach for creating multicast control trees, the Token Repository Service, is proposed, which overcomes the drawbacks of ERS and ERA.

# The Token Repository Service

## Overview and Concept

The Token Repository Service gives up the simple implementation of the previous approaches for the benefits of improved scalability and flexibility [Rothermel & Maihöfer 1999]. The core concept of the TRS approach are tokens, which are basically defined by a 3-tuple <group, owner, metrics>. When a node that can accept at most *B* children creates or joins the control tree, *B* tokens for this node are generated and stored in the TRS. *B* depends on the processing power and bandwidth capabilities of a node. The creating respectively joining node is called the tokens' owner. Each token has a metrics, which defines the quality of a token for example the height of its owner in the corresponding control tree, i.e. its distance to the root node increased by one (the root node has height 1). Parent nodes with low height in the control tree are preferable as this minimizes retransmission communication distances and delays.

When a node wants to join a given group, it requests a token belonging to this group from the TRS as the token refers to a non-occupied parent node in the control tree. The repository service selects a token of this group and hands it over to the requestor. On receiving this token the requestor can then connect to the token's owner in the corresponding control tree. Control tree nodes and therefore the reliable multicast transport protocols using the control trees have to provide an interface that allows child nodes to connect to them. Such a generic interface is described in [Kadansky et al. 2001] as a basic building block, which allows that TRS is used with every reliable multicast protocol.

When a node leaves a group, all *B* tokens belonging to this pair of group and owner are removed out of the repository. Note that the multicast transport protocol (for example TMTP or RMTP), which uses the control tree, must ensure that a leaving node has no children for example by rejoining children at other control tree nodes. The leaving node has itself allocated a token belonging to its parent in the control tree. This token is returned to the repository, which then can be reused by some other node joining this group later. The following operations summarize the interface of the Token Repository Service:

- *repCreateGroup (G, N, B)*: This operation creates a group *G* and makes it known to the repository service. The root of *G*'s control tree is identified by *N*, which has a maximum branching factor of *B*.
- *repDeleteGroup (G)*: This operation deletes all state information, mainly token information, associated with group *G* in the repository.

- *repJoinGroup(G, N, B)* returns *(Token)*: This operation is called when the node identified by *N* wants to join group *G*, where *N* has a maximum branching factor of *B*. The operation returns a token that identifies to which node *N* is supposed to connect to.
- *repLeaveGroup(G, M)*: This operation deletes all tokens owned by group member *M* for group *G* in the token repository.
- *repAddToken(G, E)*: This operation is called when a successor disconnects from a token owner *E* in *G*'s control tree. It adds a new token owned by *E* into the token repository.
- *repRefreshToken (G, E, A)*: The operation renews the token information, which is maintained according to the soft state principle. The token owner *E* announces that *A* number of tokens should still be available in the token repository.

**Realization of the Token Repository Service**

In order to create control trees which allows for high performance of the reliable multicast protocol, the following two goals should be considered: (1) Always a token with an owner that is as close as possible to the joining node should be selected (locality principle) and (2) if there are several such parents, a parent with minimal height in the control tree should be selected.

To achieve scalability, to avoid a single point of failure and to meet the locality principle, namely selecting a parent node nearby, the TRS is implemented as a distributed system consisting of token repository servers, or repServers for short. Each repServer is responsible for a disjunct set of client nodes, called domain, and is called this nodes' home repServer. A client node accesses the Token Repository Service always via its home repServer.

In principle, nodes can be arbitrarily assigned to domains and a repServer does not have to be located inside its domain. However, to construct control trees with low retransmission delays and to minimize communication overhead, domains should structure the network by communication distance. This means the communication distance between two nodes in the same domain should be typically smaller than between two nodes in different domains. Furthermore, the repServer should be located inside its domain.

Tokens are stored on the distributed repServers. Note that not all tokens of a group are stored at only one repServer, thus several repServers may store tokens for the same group and usually a repServer stores tokens of a number of groups.

If a node requests a token from its home repServer and this repServer possesses a token of the requested group, it simply delivers such a token. Since a token is always stored at the repServer responsible for its owner's domain, a repServer possesses tokens of a group only when a node in its domain has created or joined this group before. As a consequence, usually a repServer does not possess tokens for each group. Therefore, it is possible that a node's home repServer cannot satisfy a token request although another repServer could provide a suitable token.

For example, assume that all tokens of a group are stored on a single repServer $S_1$, responsible for domain 1. If a node in another domain, say domain 2 for example, requests its home repServer $S_2$ for a token, our approach must ensure that finally $S_2$ can deliver one of $S_1$'s tokens to the requesting node. To meet this requirement, a repServer initiates a token search for a group's token if a node in its domain requests a token and none is available locally. In the TRS-R strategy, a token search is processed by ERS. All repServers belong to the same well-known multicast group. If a repServer has to search for a token, it starts an ERS search on the repServers' multicast group. If a repServer receives such a token search message and possesses a token of this group, it hands over one token to the searching repServer. An ERS search on the repServers' multicast group, which is usually small, improves scalability compared with the original ERS approach, which encompass all members of the destination multicast group.

In the TRS-K and TRS-M strategy, all repServers are organized in a tree structure. Nodes access the Token Repository Service only via leaf repServers, which store the token information. Non-leaf repServers are necessary to facilitate the token search procedure, if no local token is available. Each non-leaf repServer maintains a group-specific set of all child repServers that belong to a token containing subhierarchy, called group record. A token search is processed by forwarding the search step-by-step to the parent in the repServer hierarchy, i.e. in leaf-to-root direction until a server is reached, which knows a token containing subhierarchy. Then the search is forwarded in reverse direction, i.e. in root-to-leaf direction to such a child repServer. This is repeated until a leaf repServer is reached which hands over a token to the searching node.

Instead of starting at the root node, by starting at a leaf node TRS achieves the following advantages: (1) a more local repServer can be found for delivering tokens, which results in low communication distances between child and parent nodes in the control tree, (2) lower message overhead for token searching if neighboring repServers can provide tokens, which is quite likely after the first few joins, and (3) the root repServer is no bottleneck.

While the TRS-K strategy allows only searching for an arbitrary token, the TRS-M strategy allows searching for the best available token in the TRS with respect to the tokens' metrics. To enable

this, the group record is extended so that for each subhierarchy the best metrics of all tokens in the subhierarchy is maintained. When searching a token, the extended group record can be used to find a path in the TRS hierarchy to the token with the best metrics. The simulation results show that if for example the token height is used as a metric, the resulting control trees with minimal height provide low delivery delays for the multicast transport protocol.

In summary, if a requested token is available locally at the requestor's home repServer, the requestor and the owner of this token are in the same domain. This is the best case in terms of communication overhead between the repServers and communication distance between requestor and owner in the control tree. If a token is not available locally, the search procedure tries to find a token in a domain close to the requestor's domain.

The major strength of TRS lies in its excellent scalability which will be proved in the next section. Other advantages over ERS and ERA include the independence from the network's multicast support. This means, TRS can be used in bi- and unidirectional, single- and multiple-sender, sender-based and shared-tree multicast networks with always the same performance.

## Analysis and Simulation Results

In this thesis we present numerical results from a probabilistic analysis as well as NS-2 simulation results of both reliable multicast transport protocols and the Token Repository Service.

### Reliable Multicast Transport Protocols

The probabilistic analysis shows the CPU and bandwidth requirements, the resulting throughput and the delivery delay of all introduced classes of reliable multicast transport protocols. The analysis assumes a realistic network model in which data packets and control packets can get lost and it introduces a spatial loss correlation, which is derived from measurements in the MBone.

In order to prove the analytical results, the transport protocols SRM, RMTP and TMTP have been implemented in the NS-2 simulation environment. The results of the analysis and simulations are as follows:

- Flat reliable transport protocols provide no scalability for large receiver groups and high sending rates.

- Hierarchical reliable multicast transport protocols provide scalability for large receiver groups and high sending rates.
- Sender-initiated reliable multicast transport protocols provide lower delays but results in higher network load and less scalability than receiver-initiated protocols.
- The delivery delay of receiver-initiated protocols depends on the sending rate. Low sending rates result in high delivery delays.

The following table shows the summarized results, which are proved by simulation results. As a consequence we see that hierarchical reliable multicast transport protocols are necessary to provide scalability for large receiver groups.

| protocol class | bandwidth usage | CPU usage | delivery delay | scalability |
|---|---|---|---|---|
| sender-initiated flat | $\ominus$ | $\ominus\ominus$ | $\oplus$ | $\ominus\ominus$ |
| receiver-initiated flat | $\ominus$ | $\ominus$ | $\ominus$ | $\ominus$ |
| sender-initiated hierarchical | $\oplus$ | $\oplus$ | $\oplus\oplus$ | $\oplus$ |
| receiver-initiated hierarchical | $\oplus\oplus$ | $\oplus\oplus$ | $\ominus\ominus$ | $\oplus\oplus$ |
| $\ominus\ominus$ worst | $\ominus$ bad | $\oplus$ good | $\oplus\oplus$ best | |

**Token Repository Service**

Our next focus is to compare the Token Repository Service with its alternatives ERS and ERA. The analysis and simulation studies include the message overhead that is necessary to create a control tree, the scalability for large networks and multicast groups and finally the message delivery delay in the resulting control tree. The results show the following:

- ERS and ERA provide poor scalability for large networks, large receiver groups and high background load in the network. Since control trees are only necessary for large receiver groups, ERS and ERA virtually cannot be used to create control trees.
- TRS provides excellent scalability for large networks, large receiver groups and is independent of the network load.
- In most cases, TRS provides lower message delivery delays in the resulting control trees.

- Only TRS can create control trees with large branching factors and therefore optimize the performance of reliable multicast protocols.

Recall that the branching factor is the maximum number of child nodes of a node in the control tree. Our results show, that in many cases a medium branching factor results in lower message delivery delay, lower bandwidth and CPU usage and therefore higher throughput of a reliable multicast protocol. However, such an optimized branching factor cannot be achieved with ERS and ERA, whose resulting branching factor is in most cases quite small, which limits the performance of reliable multicast protocols.

In contrast to ERS and ERA, TRS allows to use the token metrics in order to optimize the branching factor of the resulting control tree and hence the performance of reliable multicast protocols.

## Conclusion

Analysis and simulation results show that only hierarchical multicast protocols can support large multicast receiver groups. A prerequisite for using hierarchical multicast protocols is to establish the control tree, which consists of all group members.

With the Token Repository Service a new approach to create multicast control trees is presented. The former approaches, ERS and ERA suffer from several disadvantages. Although ERS provides an easy and robust way to create control trees for reliable multicast, it suffers from working only in bidirectional multicast networks and with multiple-sender routing protocols. The major disadvantage is its poor scalability proved by simulation results. ERA shares many characteristics with ERS and suffers from poor scalability, too.

The Token Repository Service (TRS) gives up the simple implementation of the previous approaches for the benefit of flexibility and most important for the benefit of improved scalability, proved by analysis and simulation results. As it relies on unicast communication to create the control tree, it can be used in every multicast network and with every multicast routing protocol. Currently, the combination of an hierarchical multicast protocol and the Token Repository Service is the only option to support large reliable multicast receiver groups.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Motivation

Das Internet hat seit seiner Entstehung im Jahre 1969 mit dem Arpanet ein stürmisches Wachstum hinter sich [Walrand & Varaiya 2000]. Das Wachstum betraf viele Bereiche wie die Anzahl der Teilnehmer, der Vermittlungsrechner und der Anwendungen gleichermaßen. Die häufigsten Anwendungen sind die elektronische Post (E-Mail) und das World Wide Web. Ein Ende dieses Wachstums ist nicht in Sicht. Firmen koppeln unterschiedliche Zweigstellen über das Internet zu einem sogenannten virtuellen privaten Netzwerk [VPNC 2001]. Neue Technologien ermöglichen den drahtlosen Zugang zum Internet von unterwegs via Handy oder Laptop [Schiller 2000]. Radio- und Fernsehprogramme können bereits über das Internet empfangen werden. Die wohl weitgehenste Veränderung wird durch „Ubiquitous Computing“ erwartet. In nicht allzu ferner Zukunft sollen Computer in jeglichen Gegenständen unseres täglichen Lebens ihre Arbeit verrichten und dabei eine ständige Verbindung zum Internet haben [Weiser 1991, Weiser 1993]. Alle diese Entwicklungen haben eines gemeinsam. Sie bedingen einen Ausbau der Internet-Bandbreite oder zumindest eine effizientere Nutzung der vorhandenen. Mit Multicast bezeichnet man die möglichst bandbreitenschonende Auslieferung einer Nachricht an eine Gruppe von Empfängern. Multicast kann für Gruppenkommunikation die verfügbare Bandbreite effizienter nutzen als die herkömmliche Punkt-zu-Punkt Kommunikation.

Beispiele für Anwendungen, in denen Multicast gewinnbringend eingesetzt werden kann, lassen sich unschwer ausmachen. Dazu zählen Informationsverteilungen wie Aktienkurse, Softwareverteilung und inhärent verteilte Anwendungen wie Audio– und Videokonferenzen, „Groupware“, verteiltes Rechnen und verteilte Spiele. Ein Beispiel für ein Produkt ist das Pointcast-Netzwerk [Ramakrishnan & Dayal 1998]. Ein Kunde abonniert verschiedene für ihn interessante Kanäle, beispielsweise politische Ereignisse oder Finanznachrichten und bekommt in dessen Folge die gewünschte Information automatisch geliefert (Push-Kommunikation). Leider verwendet Pointcast, aufgrund der noch

nicht flächendeckenden Verfügbarkeit von IP-Multicast, nur herkömmliche und für diese Zwecke ineffiziente Punkt-zu-Punkt Kommunikation für die Auslieferung von Nachrichten. Dies führte in einer Studie aus dem Jahre 1997 dazu, dass Pointcast, obwohl es deutlich weniger Benutzer als andere Informationsdienste aufweisen konnte, für den meisten Internet-Verkehr verantwortlich war (mehr als 17% des Gesamtverkehrs entfielen auf Pointcast, dicht gefolgt von Zugriffen auf das Netscape Portal, alle anderen, z.B. Microsoft, Yahoo oder CNN, verzeichnen einen deutlichen Abstand) [Graham-Cumming 1997]. Nachdrücklich zeigt sich hier, dass eine bandbreitenschonende Auslieferung durch einen Multicast-Dienst dringend erforderlich ist.

Durch Multicast kann auch die Qualität oder Fairness eines Dienstes als positiver Nebeneffekt verbessert werden. Bei einer Videokonferenz mit vielen Teilnehmern wirkt sich das gleichzeitige anstatt sequentielle Senden der Bildinhalte an alle Teilnehmer in einer geringeren Verzögerung und damit gesteigerten Qualität aus. Bei verteilten Spielen kann die Fairness zwischen den Teilnehmern gesteigert werden, da zumindest das Senden einer veränderten Situation an alle gleichzeitig erfolgt. Dies gilt noch verstärkt für elektronische Märkte, da hier erhebliche finanzielle Auswirkungen zu bedenken sind. Wird beispielsweise Aktienhandel über das Internet betrieben, so ist die Zustellung der Kursinformationen sowie der Kauf– und Verkaufsangebote mit Vorteilen für denjenigen verbunden, der diese Information als erster erhält und schnellstmöglich darauf reagieren kann.

Im Rahmen dieser Arbeit wird ein zuverlässiger Multicast-Dienst betrachtet. Zuverlässig bedeutet, dass die Auslieferung an alle Teilnehmer garantiert wird, was durch Übertragungswiederholungen bei einem Nachrichtenverlust erreicht wird. Die Garantie der Auslieferung ist indessen an bestimmte Voraussetzungen geknüpft, auf die im Verlauf der Arbeit näher eingegangen wird. Eine zuverlässige Nachrichtenauslieferung wird für viele Anwendungen verlangt, z.B. Informations– und Softwareverteilung, „Groupware“, verteiltes Rechnen und verteilte Spiele. Auch Anwendungen bei denen vorrangig die schnelle Auslieferung einer Nachricht und nicht deren garantierte Auslieferung notwendig ist, können von einer zuverlässigen Auslieferung profitieren. Dazu gehört die oftmals in diesem Zusammenhang wenig beachtete kodierte Audio– und Videoübertragung, die oftmals nur mittels zusätzlicher Redundanz (forward error correction, FEC) zwar eine höhere Übertragungssicherheit erlangt aber keine wirklich zuverlässige Auslieferung. Kodierte Videoübertragung nach dem MPEG-Standard reagiert jedoch mit sichtbaren Bildstörungen auf den Verlust bestimmter Informationen, den sogenannten I-Rahmen, was FEC nicht gänzlich verhindern kann. Vereinfacht ausgedrückt enthalten I-Rahmen ein vollständiges Bild, während die nachfolgenden Bilder als Differenz zu diesem Bild übertragen werden. Fehlt ein I-Rahmen, so müssen viele Folgebilder mangels Bezugspunkt verworfen werden, was die Bildqualität deutlich beeinträchtigt. Unter der Voraussetzung, dass eine kurzzeitige Pufferung der Video– und Audiodaten akzeptabel ist, profitiert die Bild– und Tonqualität erheblich von Übertragungswiederholungen verlorener Daten. Die Ergebnisse in [Pejhan et al. 1996] belegen, dass mit zuverlässigem Multicast übertragene I-Rahmen die Bildqualität gegenüber FEC merklich verbessern. Für die Differenzbilder ist eine unsichere Übertragung ausreichend.

Die Verwendung eines zuverlässigen Multicast-Protokolls besitzt einen weiteren positiven Nebeneffekt. Die Implementierung von Fluss– und Überlastkontrolle ist wesentlich einfacher zu realisieren, da die notwendigen Kontrollnachrichten zur Erkennung von Nachrichtenverlust auch dafür verwendet werden können. Die Fluss– und Überlastkontrolle ist eine Voraussetzung dafür, Multicast in großem Umfang einzusetzen, da nur dadurch die Fairness zwischen TCP– und Multicast-Verkehr gewährleistet werden kann.

## 1.2 Beitrag und Aufbau der Arbeit

Um einen zuverlässigen Multicast-Dienst für eine große Teilnehmerzahl zu realisieren wurden in der Literatur hierarchische Transportprotokolle basierend auf IP-Multicast vorgeschlagen. Hierarchische Protokolle erreichen die notwendige Skalierbarkeit, d.h. die Fähigkeit auch mit zunehmenden Mitgliederzahlen effizient zu arbeiten, indem sich alle Gruppenmitglieder die Last der Übertragungswiederholungen in einer hierarchischen Organisation teilen.

Eine Vorbedingung für den Einsatz hierarchischer Protokolle ist damit aber durch die Notwendigkeit gegeben, eben diese Hierarchie aufzubauen. Die Untersuchung herkömmlicher Verfahren in Kapitel 3 zeigt, dass diese lediglich für kleine Gruppen einsetzbar sind, da sie nicht mit steigender Mitgliederzahl skalieren und weitere gewichtige Nachteile haben. Als ein wesentlicher Beitrag dieser Arbeit wird ein neues Verfahren zum Aufbau der Hierarchie, genannt Token-Repository-Service, vorgestellt. Der Token-Repository-Service vermeidet die Nachteile bisheriger Verfahren und kann universell eingesetzt werden. Erst damit können hierarchische Protokolle und folglich ein zuverlässiger Multicast-Dienst für große Teilnehmerzahlen erfolgreich etabliert werden. In umfangreichen Simulationen wird der Token-Repository-Service mit bisherigen Verfahren verglichen und sowohl die Skalierbarkeit als auch weitere quantifizierbare Vorteile unter Beweis gestellt.

Für zuverlässige Multicast-Dienste existieren bereits ein Reihe von Transportprotokollen mit unterschiedlichen Stärken und Schwächen. Diese Stärken und Schwächen können sowohl qualitativer als auch quantitativer Natur sein. Qualitative Kriterien umfassen beispielsweise den Grad der garantierten Zuverlässigkeit, die Ordnungserhaltung bei der Nachrichtenauslieferung und die Verträglichkeit mit herkömmlichen Routing-Protokollen. Qualitative Kriterien werden im Rahmen dieser Arbeit nicht vertiefend verfolgt, da die verschiedenen Protokolle dazu äußerst unterschiedliche Ansätze verfolgen und oftmals sehr ungenau beschrieben sind. Eine kurze Diskussion befindet sich dazu in den Abschnitten 2.5.1, 2.5.2, 4.2 sowie in [Bäurle 1999].

Quantitative Eigenschaften zuverlässiger Multicast-Transportprotokolle umfassen unter anderem die CPU-Belastung, den Nachrichtenaufwand und die Verzögerung der Nachrichtenauslieferung. Diese Kriterien geben einen Hinweis auf die Skalierbarkeit eines Protokolls auch für große Empfängergruppen. Unter Zuhilfenahme dieser Kriterien kann eine Auswahl des geeigneten Protokolls für den

gewünschten Einsatzzweck erfolgen. Quantitative Kriterien verschiedener Protokolle wurden bisher noch nicht ausreichend untersucht. In Kapitel 4 werden die aufgeführten quantitativen Kriterien im Rahmen einer probabilistischen Analyse untersucht. Grundlage der Analyse ist eine sinnvolle Klassifizierung der gebräuchlichsten Verfahren, um eine zuverlässige Nachrichtenzustellung zu realisieren. Die mathematische Aufarbeitung wird durch anschauliche numerische Ergebnisse über die wichtigsten Erkenntnisse abgeschlossen.

Ein zentrales Ergebnis der Analyse ist, dass die Klasse der hierarchischen Transportprotokolle bedeutsame Vorteile für sich verbuchen kann und schließlich die einzige Protokollklasse ist, die auch für größte Empfängergruppen skaliert. Gerade dies ist entscheidend für Multicast-Protokolle, da die prinzipiellen Vorteile der Verwendung von Multicast mit zunehmender Gruppengröße ebenfalls zunehmen. Umfangreiche Simulationen bestätigen die Ergebnisse der Analyse und zeigen damit die Überlegenheit hierarchischer Protokolle. Da der Token-Repository-Service eine Voraussetzung für den Einsatz hierarchischer Transportprotokolle darstellt, ist damit dessen Notwendigkeit belegt. Schließlich wird die Arbeit mit einer Zusammenfassung und einem Ausblick abgeschlossen.

# Kapitel 2

# Grundlagen der Multicast-Kommunikation

## 2.1 Überblick

In der Einleitung wurde Multicast als die bandbreitenschonende Auslieferung einer Nachricht an eine Gruppe von Empfängern definiert. Nach der Definition in [JTC1/SC6 1994] erlaubt Multicast im engeren Sinne nur einen Sender. Sind beliebig viele Sender zulässig, so spricht man von Multipeer-Kommunikation. Diese Definition wird allerdings nicht von allen Autoren verwendet. Zum Beispiel kann IP-Multicast, abhängig vom verwendeten Routing-Protokoll, durchaus ein Multipeer-Dienst sein. Im weiteren Verlauf dieser Arbeit wird daher wie gebräuchlich Multicast als die Verallgemeinerung dieser beiden Varianten verwendet. Ist die Unterscheidung notwendig, so wird gesondert darauf hingewiesen.

Die einfachste Möglichkeit, in einem Netzwerk einen Multicast-Dienst zu realisieren, besteht darin, eine Nachricht an alle Rechner zu senden. Sind Rechner nicht am Empfang interessiert, sind sie selbst dafür verantwortlich die nicht relevanten Nachrichten auszufiltern. Das Ausliefern an alle Rechner wird als Broadcast bezeichnet. Multicast beinhaltet folglich ein Spektrum mit Broadcast und Unicast, dem Senden einer Nachricht an genau einen Empfänger, als Extreme. Zur vollständigen Klassifizierung sei hier noch Anycast [Partridge et al. 1993] genannt. Anycast bezeichnet die Auslieferung an einen beliebigen Empfänger aus einer Gruppe möglicher Empfänger.

Die oben aufgezeigte Realisierungsvariante für einen Multicast-Dienst mittels Broadcast hat einen schwerwiegenden Nachteil, der einen Einsatz in großem Umfang verbietet. Diese Lösung bietet keine Effizienz und damit keine Skalierbarkeit, da das Netzwerk mit unnötigen Nachrichten belastet wird. Mit der Effizienz wird zugleich auch der wichtigste Grund angesprochen, warum ein Multicast-Dienst überhaupt notwendig ist.

Im Folgenden werden die Grundlagen der tatsächlichen Realisierung des Multicast-Diensts im Internet gelegt. Dabei unterscheidet man zwischen Multicast innerhalb eines lokalen Netzes, welcher

nach dem ISO-OSI-Referenzmodell [Day & Zimmerman 1983] auf der Sicherungsschicht angesiedelt ist und Multicast in Weitverkehrsnetzen, welcher auf der Vermittlungsschicht abläuft. Zusammenfassend wird dies als IP-Multicast bezeichnet. Der erste Teil dieses Grundlagenkapitels widmet sich der Einführung wesentlicher Konzepte von IP-Multicast. Zwei wichtige Gründe sprechen dafür. In dieser Arbeit betrachtete zuverlässige Multicast-Protokolle arbeiten auf der Transportschicht aufbauend auf IP-Multicast als unzuverlässigem Basisdienst. Desweiteren wird das Verständnis der hier eingeführten Multicast-Routing-Verfahren wesentliche Voraussetzung für die Beurteilung und Interpretation der Analyse– und Simulationsergebnisse sein. Der zweite Teil dieses Kapitels widmet sich unter der Annahme, dass IP-Multicast verfügbar ist, den zuverlässigen Multicast-Protokollen auf Transportschicht. Für eine Auswahl der besprochenen Multicast-Transportprotokolle werden in Kapitel 4 Simulationsresultate vorgestellt.

## 2.2 IP-Multicast Dienstmodell

Im Gegensatz zur Unicast-Kommunikation muss bei einem Multicast-Dienst nicht nur ein Empfänger, sondern eine Menge von Empfängern adressiert werden. Ein naiver Lösungsansatz wäre die Adressierung über eine Empfängerliste, die bei jeder Nachricht mitgesendet werden muss. Dies würde allerdings einen erheblichen Aufwand bei den Vermittlungsrechnern verursachen, da diese die Nachrichten aufwändig verarbeiten und neu zusammenstellen müssten. Eine andere Lösung besteht darin, die Nachrichten an eine spezielle Gruppenadresse zu senden. Der Gruppenadresse ist nicht anzusehen, welche und wie viele Mitglieder sie beinhaltet. Dies ist eine wichtige Eigenschaft des IP-Multicast-Dienstmodells wie es im Internet realisiert wurde [Cheriton & Deering 1985, Deering & Cheriton 1985, Deering 1989]. Eine sogenannte Host-Gruppe umfasst eine beliebige Mitgliederzahl, die alle über dieselbe Adresse erreicht werden können. Vom Sender aus gesehen gibt es daher keinen Unterschied zum Senden einer Unicast-Nachricht.

Eine Gruppe ist nicht statisch, vielmehr können zu beliebigen Zeitpunkten Mitglieder eine Gruppe verlassen und andere Mitglieder eintreten. Eine wichtige Eigenschaft des Dienstmodells ist, dass die Gruppenverwaltung nicht zentral, z.B. über den Sender abläuft, sondern dass dies von den Empfängern aus geschieht. Die Empfänger sind anonym, d.h. dem Sender nicht bekannt. Damit sind allein die Vermittlungsrechner für die Auslieferung an alle Teilnehmer zuständig und nicht der Sender. Eine Garantie für die Auslieferung wird indessen nicht übernommen. Dies ist höheren Protokollschichten vorbehalten. Das Dienstmodell besagt auch, dass der Sender nicht zwingend Mitglied der Gruppe sein muss, an die er sendet.

## 2.3 IP-Multicast in lokalen Netzen

Im folgenden Abschnitt werden die Abläufe von IP-Multicast in lokalen Netzen erörtert. Zu den wesentlichen Aufgaben zählt der Beitritt eines Rechners zu einer Gruppe sowie das Verlassen einer Gruppe, was unter dem Begriff Gruppenverwaltung zusammengefasst wird. Daneben ist das eigentliche Zustellen, d.h. Senden und Ausliefern von Paketen, von Bedeutung. Folgende Bücher geben einen guten Überblick über diese Thematik: [Meyer 1998, Kosiur 1998, Maufer 1997, Wittmann & Zitterbart 1999, Miller 1998].

### 2.3.1 Überblick

Für den Beitritt eines Endsystems[1] zu einer IP-Multicast-Gruppe sind folgende Schritte notwendig:

- IP-Multicast-Adresse ermitteln
- Abbildung der IP-Multicast-Adresse auf die MAC (Medium Access Control) Multicast-Adresse des LANs (Local Area Network) durchführen
- Netzwerkadapter für den Empfang der Gruppe konfigurieren
- Lokalen Router (engl. Designated Router, DR) mittels IGMP (Internet Group Management Protocol) über den Gruppenbeitritt informieren

Das Ermitteln der IP-Multicast-Adresse ist applikationsspezifisch. Eine generelle Möglichkeit stellt das Session-Directory (SDR) dar [Handley & Jacobson 1998, Handley et al. 1999, Handley et al. 2000]. Eine weitere Möglichkeit sind applikationsspezifisch reservierte Multicast-Adressen.

Das Verlassen einer Gruppe erfolgt indem das Ende der Mitgliedschaft dem lokalen Router angezeigt wird und dieser die Paketweiterleitung daraufhin einstellt. In den folgenden Abschnitten werden diese Schritte detailliert dargestellt.

### 2.3.2 Adressabbildung

Im IP-Protokoll ist ein eigener Adressbereich für Multicast-Kommunikation reserviert, die Klasse-D-Adressen. Die nach IEEE 802 standardisierten LANs wie Ethernet oder Token Ring, verfügen über

[1] Der Begriff *Endsystem* kennzeichnet Rechner mit den ISO-OSI-Schichten ein bis sieben. Demgegenüber besitzt ein *Router* nur die ISO-OSI-Schichten eins bis drei.

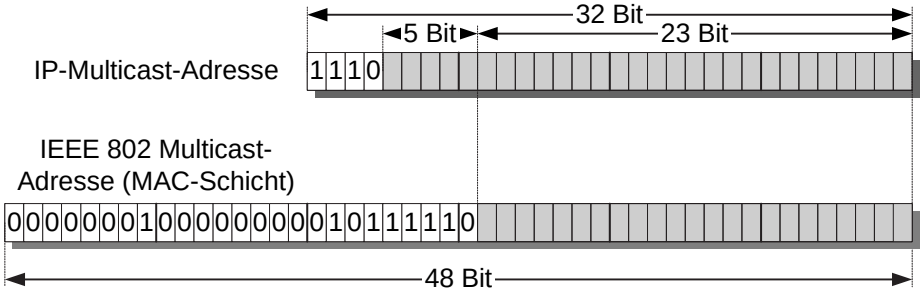



*Abbildung 2.1: Adressabbildung einer IP-Multicast-Adresse auf eine MAC-Multicast-Adresse*

einheitliche MAC-Adressen mit einer Länge von 48 Bit. Darin sind ebenfalls Multicast-Adressen vorgesehen, d.h. der Netzwerkadapter kann für den Empfang von Multicast-Paketen konfiguriert werden. Diese MAC-Adressen können allerdings leider nicht gleich gewählt werden wie die IP-Multicast-Adressen. Für eine IP-Multicast-Adresse stehen 28 Bit zur freien Verfügung (dies entspricht ungefähr 268 Millionen Adressen), während für eine MAC-Multicast-Adresse lediglich 23 Bit zur freien Verfügung stehen [IEEE 2000]. Die Lösung besteht darin, dass nur die unteren 23 Bit der IP-Multicast-Adresse auf die MAC-Adresse abgebildet werden. Die oberen 5 Bit bleiben unbeachtet (siehe Abbildung 2.1). Daraus resultiert, dass die Abbildung nicht eindeutig ist, d.h. $2^5 = 32$ IP-Multicast-Gruppen werden auf eine MAC-Multicast-Gruppe abgebildet. Für einen Empfänger kann dies bedeuten, dass er auch Rahmen[2] von Gruppen empfängt, die er nicht abonniert hat. Die zuviel empfangenen Rahmen werden jedoch nicht an die Applikation weitergereicht, sondern bereits im Protokollstapel durch Vergleich der IP-Adressen verworfen.

### 2.3.3 Gruppenbeitritt und –austritt

Nachdem der Netzwerkadapter, hier soll beispielhaft Ethernet verwendet werden, für den Empfang der ermittelten Multicast-Adresse konfiguriert wurde, muss der lokale Router über den Gruppenbeitritt informiert werden. Dazu wird das Internet-Group-Management-Protocol (IGMP, siehe auch Glossar im Anhang) verwendet. IGMP ist ein integraler Bestandteil von IP und hat die Aufgabe, die Gruppenverwaltung zu ermöglichen, d.h. das Beitreten zu und Verlassen von Gruppen. Da IGMP nur im lokalen Netz eine Bedeutung besitzt, werden IGMP-Pakete von den Routern nicht weitergeleitet. Von IGMP existieren derzeit drei Versionen. Zunächst soll die Version 1 besprochen werden [Deering 1989] und anschließend die Neuerungen von Version 2 [Fenner 1997] und 3 [Cain et al. 1999].

Der Beitritt zu einer Gruppe wird von einem Endsystem durch ein IGMP-Report-Paket dem lokalen Router angezeigt. Das IGMP-Report-Paket wird an die gewünschte Gruppe adressiert. Der lokale

[2] Dateneinheiten werden auf der Sicherungsschicht als Rahmen, auf der Vermittlungsschicht als Pakete und auf der Transportschicht und allen höheren Schichten als Nachrichten bezeichnet [Tanenbaum 1997].

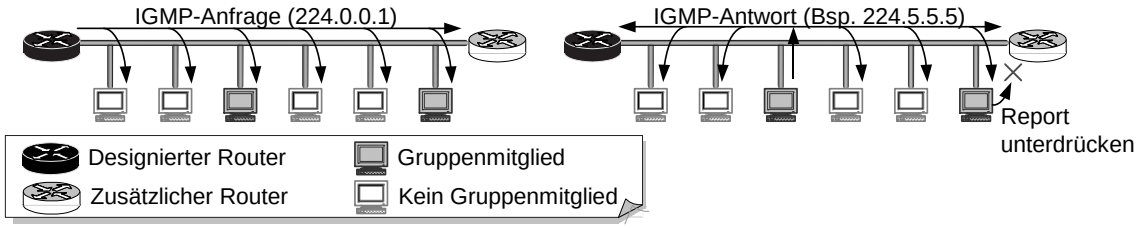



*Abbildung 2.2: Gruppeneintritt in lokalen Netzen*

Router empfängt dazu die Pakete für alle Multicast-Adressen. Nachdem ein Router von der Gruppenmitgliedschaft erfährt, leitet er eingehende IP-Multicast-Pakete in das lokale Netz weiter.

Neben dem Empfänger kann auch der lokale Router als Initiator des Gruppenbeitritts dienen (siehe Abbildung 2.2). Der lokale Router sendet periodisch IGMP-Anfragen an alle Rechner im LAN (all-hosts-group, 224.0.0.1). Ein Teilnehmer, der die Anfrage erhält und einer Gruppe angehört oder beitreten möchte, antwortet mit einem IGMP-Report-Paket wie oben beschrieben. Ist der Teilnehmer in mehreren IP-Multicast-Gruppen, so muss für jede Gruppe ein Report-Paket gesendet werden. Da der Router nur wissen muss, ob mindestens ein Gruppenmitglied in diesem LAN vertreten ist, ist ein IGMP-Report-Paket pro Gruppe ausreichend. Deshalb verzögert jeder Teilnehmer seine Antwort um eine zufällige Zeit und unterlässt das Senden, wenn in der Zwischenzeit ein Report-Paket für diese Gruppe von einem anderen Teilnehmer empfangen wurde.

Für das Verlassen einer Gruppe gibt es keine explizite Möglichkeit in IGMPv1. Ein Teilnehmer muss warten bis die nächste IGMP-Anfrage des lokalen Routers empfangen wird. Anstatt nun mit einem IGMP-Report zu antworten wird einfach nicht geantwortet. Antwortet auch kein anderer Teilnehmer, so wird der Router das Weiterleiten von IP-Multicast-Paketen für diese Gruppe einstellen.

### 2.3.4 Neuerungen von IGMPv2 und IGMPv3

Der offensichtlichste Nachteil von IGMPv1 ist die fehlende Möglichkeit eine Gruppe explizit zu verlassen, was letztendlich mehr Verkehr im lokalen Netz und dadurch Bandbreitenverschwendung bedeutet. In IGMPv2 wurde daher die Möglichkeit vorgesehen, eine Gruppe durch ein Austrittspaket an den lokalen Router schnell zu verlassen [Fenner 1997].

Ein weiterer Nachteil ist nicht so offensichtlich aber dennoch ein Problem für den Einsatz von Gruppenkommunikation. Wenn eine Gruppe etabliert wurde, können prinzipiell beliebige Rechner Nachrichten an diese Gruppe senden.[3] Diese Eigenschaft kann natürlich sehr einfach missbraucht werden, um unerwünschte Nachrichten an eine möglicherweise sehr große Empfängergruppe zu senden. In

[3]Erschwerend kommt hinzu, dass bei manchen Routing-Verfahren der sendende Rechner nicht einmal zwingend zur Gruppe gehören muss.

IGMPv3 ist vorgesehen, dass eine Anmeldung nicht nur gruppenspezifisch erfolgt, sondern auch senderspezifisch sein kann. Auch ist es möglich eine Negativliste zu erstellen. Diese beinhaltet alle Sender, von denen ein Empfänger keine Nachrichten erhalten möchte. Bisher ist IGMPv3 jedoch nur als Arbeitsdokument verfügbar und die Standardisierung noch nicht abgeschlossen [Cain et al. 1999].

## 2.4 IP-Multicast in Weitverkehrsnetzen

Nachdem im vorigen Abschnitt das Gruppenmanagement und die Zustellung von Multicast-Paketen im lokalen Netz besprochen wurde, wird nun die Situation im Weitverkehrsbereich erörtert. In Abschnitt 2.4.1 werden die verschiedenen Routing-Verfahren eingeführt. Darauf wird später Bezug genommen, da sie eine wichtige Grundlage für das Verständnis des Token-Repository-Service darstellen und Voraussetzung für die Interpretation der Analyse– und Simulationsresultate sind. Schließlich wird in Abschnitt 2.4.2 dargelegt wie gegenwärtig die IP-Multicast-Struktur im Internet aussieht.

### 2.4.1 Routing-Verfahren

Das Grundproblem des Multicast-Routings stellt sich folgendermaßen dar: Gegeben sind ein oder mehrere Sender und eine Menge von Empfängern. Pakete, die von einem Sender an die Gruppe adressiert werden, müssen an wenigstens (oder genau) diese Menge von Empfängern propagiert werden. Es wurden in der Vergangenheit mehrere Routing-Protokolle spezifiziert, die neben der korrekten Paketauslieferung spezifische Eigenschaften bezüglich der Verwendung von Unicast-Protokollen, Bandbreitenverbrauch, Speicherverbrauch auf den Routern, usw. aufweisen. Im Folgenden sollen die wichtigsten Protokolle, DVMRP, PIM-SM und PIM-DM, CBT sowie MOSPF, vorgestellt werden.

#### 2.4.1.1 Grundlagen

Die Routing-Protokolle lassen sich in zwei grundsätzlich verschiedene Klassen einteilen: die senderbasierten Verfahren und die rendezvousbasierten Verfahren. Ein senderbasiertes Routing-Protokoll baut für jeden Sender einen eigenen Spannbaum für die Paketauslieferung auf, der alle Empfänger enthält. In Bild 2.3a ist dies dargestellt. Sowohl Endsystem *M1* als auch Endsystem *M2* sind potentielle Sender, weshalb die Router für beide Sender die Spannbäume ermitteln und speichern.

Im Gegensatz dazu wird bei einem rendezvousbasierten Verfahren nur ein gemeinsamer Spannbaum für alle Sender ermittelt. Alle Kanten dieses Spannbaums können dabei prinzipiell in beide Richtungen verwendet werden wie Bild 2.3b anschaulich darlegt. Der Spannbaum wird von einem Rendezvous-Router beginnend aufgebaut, der im einfachsten Fall statisch bestimmt wird.

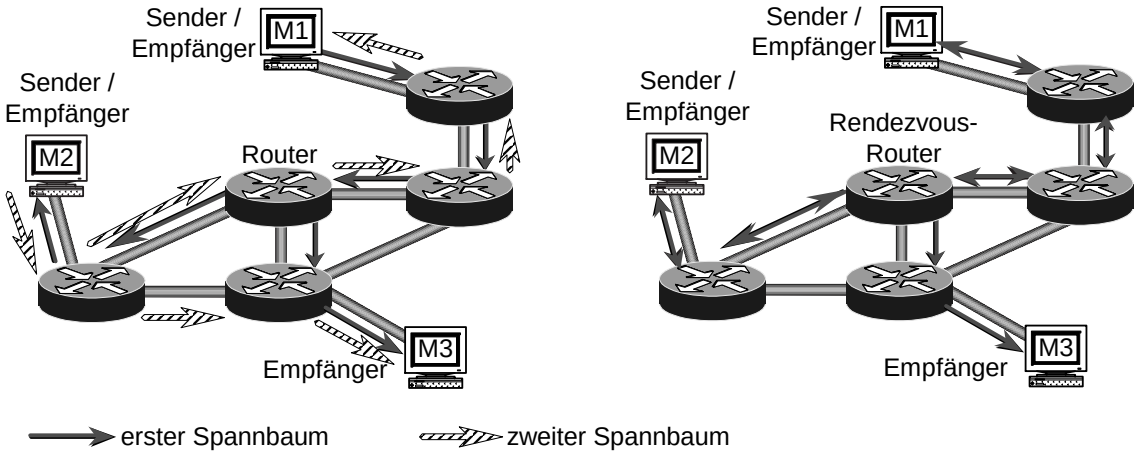



*Abbildung 2.3: Vergleich von a) senderbasiertem Routing und b) rendezvousbasiertem Routing*

Der Hauptvorteil des senderbasierten Verfahrens ist die aus Empfängersicht optimale Auslieferung einer Nachricht. Wenn beispielsweise der Sender *M2* eine Nachricht an die Gruppe sendet, so muss diese zum Erreichen von *M3* nur zwei Router passieren. Beim Vergleich mit dem rendezvousbasierten Verfahren in Bild 2.3 erkennt man, dass in diesem Falle die Nachricht auch den Rendezvous-Router passiert und damit insgesamt drei Router.

Der wesentliche Nachteil des senderbasierten Verfahrens sind die hohen Kosten zum Aufbau eines individuellen Spannbaums für jeden Sender und die Kosten zum Speichern dieser Spannbäume auf den Routern. Sei $G$ die Anzahl der Multicast-Gruppen und $S$ die Anzahl der Sender pro Multicast-Gruppe. Die zu speichernde Informationsmenge pro Router beim senderbasierten Verfahren ist damit in der Größenordnung $O(G \cdot S)$. Beim rendezvousbasierten Verfahren wird pro Gruppe, unabhängig von der Anzahl der Sender, nur ein Spannbaum aufgebaut. Folglich sind die Kosten zur Speicherung der Spannbäume pro Router lediglich in der Größenordnung $O(G)$.

Eine weitere Einteilung der Routing-Protokolle erfolgt anhand ihrer Eignung für verschiedene Gruppendichten. Man unterscheidet dabei zwischen dichten Gruppen und verstreuten Gruppen. Eine dichte Gruppe liegt vor, wenn verhältnismäßig viele Endsysteme in einem Netzwerk zur Multicast-Gruppe gehören. Dies ist häufig der Fall, wenn die Multicast-Gruppe regional, z.B. auf ein Firmennetzwerk, beschränkt ist. Hingegen kann bei weltweiten Gruppen aufgrund der großen Anzahl von Routern im Vergleich zu Gruppenmitgliedern überwiegend von weitverstreuten Gruppenmitgliedern ausgegangen werden. Die Routing-Protokolle unterscheiden sich grundsätzlich für beide Szenarien.

Bei einer dichten Gruppe kann es akzeptabel sein, wie bereits in Abschnitt 2.1 angesprochen, eine Broadcast-Auslieferung durchzuführen. Man nennt dieses Senden an alle Rechner im Netzwerk auch Fluten (engl. flooding). Dies bringt mehrere Vorteile mit sich. So ist die Gruppenverwaltung sehr einfach, da die Gruppenmitgliedschaft den Routern nicht bekannt sein muss. Desweiteren ist das

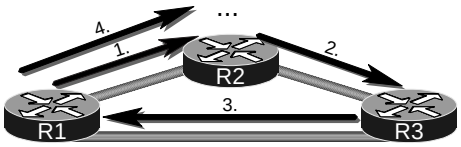



*Abbildung 2.4: Entstehung von Zyklen durch naives Fluten*

Weiterleiten der Pakete vereinfacht, da kein Spannbaum bei den Routern hinterlegt sein muss. Reale Protokolle für dichte Gruppen sind auf Algorithmen zum Fluten des Netzwerks aufgebaut, versuchen jedoch gleichzeitig fluten möglichst nicht bei jeder Paketauslieferung durchzuführen, um die Bandbreite effizienter zu nutzen. Daher besitzt der zuletzt genannte Vorteil nur beschränkte Gültigkeit für reale Protokolle, da diese dennoch Spannbäume auf den Routern verwalten.

Für verstreute Gruppen ist das Fluten des Netzwerks nicht akzeptabel, da die Bandbreitenverschwendung enorm wäre. Protokolle für verstreute Gruppen bauen daher auf andere Verfahren auf. Allen ist gemeinsam, dass das Gruppenmanagement nun explizit erfolgen muss und ein Spannbaum erstellt wird, der nur die Gruppenmitglieder und notwendige Router umfasst. Das heißt, der lokale Router, der mittels IGMP über den Gruppenbeitritt eines Hosts in seinem LAN informiert wurde, muss nun dafür Sorge tragen, in den bzw. die Spannbäume für die Paketauslieferung aufgenommen zu werden. Entsprechend erfolgt auch das Verlassen des Spannbaums explizit, um Bandbreite zu schonen.

In den folgenden Abschnitten werden die wichtigsten Multicast-Routing-Protokolle vorgestellt.

#### 2.4.1.2 DVMRP

Das Distance-Vector-Multicast-Routing-Protocol (DVMRP) [Waitzman et al. 1988] gehört zur Klasse der senderbasierten Protokolle, d.h. für jeden Sender wird ein eigener, optimierter Spannbaum aufgebaut. Desweiteren ist DVMRP für dichte Gruppen optimiert. Die Basis für DVMRP ist das Reverse-Path-Forwarding-Protokoll (RPF) [Dalal & Metcalfe 1978, Cheriton & Deering 1985], um das Netzwerk zu fluten. Nachdem das Netzwerk geflutet wurde, werden die unnötigen Pfade, d.h. die Pfade auf denen kein Empfänger liegt, abgeschnitten (engl. pruning), so dass für die nachfolgenden Multicast-Pakete ein minimaler Spannbaum zur Auslieferung verwendet wird. Nach einer bestimmten Zeit wird der Vorgang aus Fluten und Zurückschneiden wiederholt. In RFC 1075 [Waitzman et al. 1988] ist eine Minute bis zum erneuten Fluten definiert.

Um das Netzwerk zu fluten, würde ein naiver Algorithmus folgendermaßen aussehen. Ein Router leitet ein empfangenes Paket an alle Ausgänge weiter, abgesehen von dem, auf dem das Paket empfangen wurde. Dieser naive Ansatz ist indessen weder optimal noch korrekt. Abbildung 2.4 zeigt, dass dabei Zyklen entstehen können, d.h. der Algorithmus terminiert nicht.

RPF ist ein korrekter Algorithmus zum Fluten des Netzwerks, der gleichzeitig unnötige, doppelte Auslieferungen minimiert. Dafür sind jedoch Kenntnisse über die Unicast-Routing-Tabellen notwendig. Jeder Router arbeitet dabei folgenden Algorithmus beim Empfang eines Pakets ab:

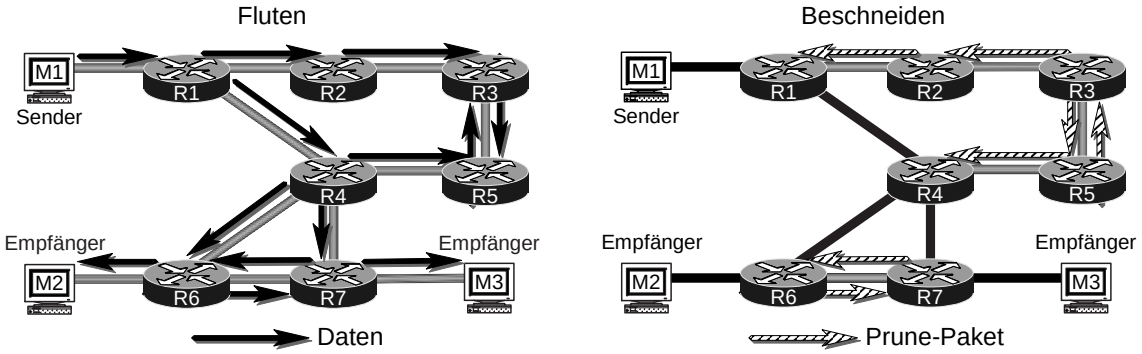



*Abbildung 2.5: DVMRP: a) Fluten des Netzwerks mit RPF und b) Beschneiden des Spannbaums durch Prune-Pakete*

1. Wenn ein Multicast-Paket über die Verbindung empfangen wurde, über die auch Unicast-Pakete zu seinem Sender gesendet werden, dann leite das Paket an alle weiteren Verbindungen durch.

2. Ansonsten verwerfe das Paket, da dieses Paket bisher nicht den besten Weg genommen hat und folglich ein unnötiges Duplikat ist. Dies verhindert die Entstehung von Zyklen.

Abbildung 2.5a zeigt die gesendeten Pakete bei Verwendung von RPF. Zu beachten ist hier, dass Router *R5* im Gegensatz zum naiven Algorithmus bei RPF das Paket von *R3* nicht weiterleitet, da dieses nicht auf dem Pfad empfangen wurde, auf dem *R5* an *M1* senden würde, nämlich *R4*. Das Paket von *R4* wird an *R3* weitergeleitet, aber bei *R3* als unnötiges Duplikat erkannt und verworfen. Somit wird die Entstehung von Zyklen unterbunden.

Nachdem das Netzwerk mit RPF geflutet wurde, werden die unnötigen Pfade abgeschnitten, damit ein minimaler Spannbaum entsteht. Unnötige Pfade sind sowohl Pfade, auf denen kein Mitglied der Multicast-Gruppe erreicht werden kann als auch Pfade, auf denen aufgrund der Unicast-Routing-Tabellen keine Pakete zum Sender gesendet werden. Der vollständige DVMRP-Algorithmus arbeitet wie folgt:

1. Der Multicast-Sender sendet das erste Paket mit RPF an alle Router im Netzwerk.

2. Empfängt ein Router ein Paket, so überprüft er, ob das Paket auf der Verbindung des kürzesten Pfads zum Sender empfangen wurde.

   (a) Ist dies der Fall, dann leite das Paket an alle anderen Verbindungen weiter, die noch nicht abgeschnitten wurden.

   (b) Ist dies nicht der Fall, dann verwerfe das Paket und antworte mit einem Prune-Paket.

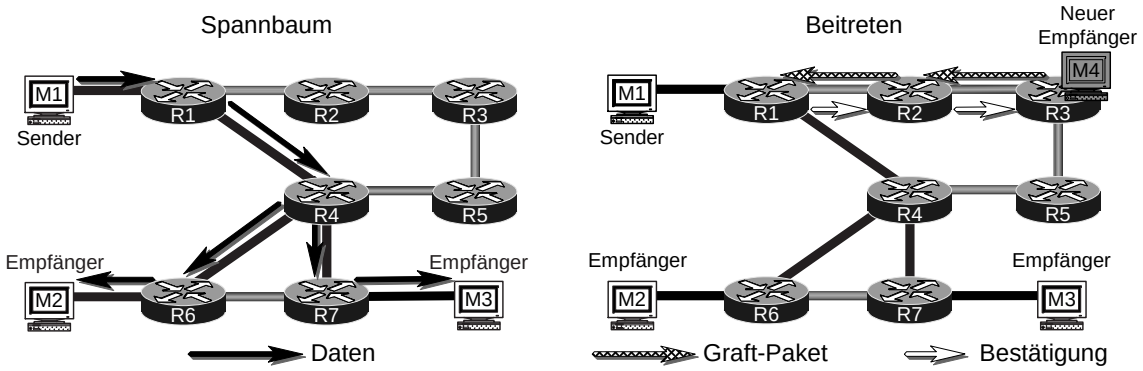



*Abbildung 2.6: DVMRP: a) Paketauslieferung über den Spannbaum und b) Eingliederung eines weiteren Teilnehmers durch Graft-Pakete*

3. Trifft ein Multicast-Paket bei einem Blatt-Router ein, d.h. bei einem Router der nur diese eine Verbindung besitzt und dieser Router hat in seinem angeschlossenen LAN keine Multicast-Mitglieder dieser Gruppe, dann antworte mit einem Prune-Paket.

4. Falls ein Router von jedem Nachfolger ein Prune-Paket erhält, so wird ebenfalls ein Prune-Paket an den Vorgänger, d.h. in Richtung Multicast-Sender, weitergeleitet.

Anhand von Abbildung 2.5b soll dies näher erläutert werden. Ein Blatt-Router ist in diesem Beispiel nicht vorhanden, allerdings der mehrfache Paketempfang. *R3* und *R5* senden sich gegenseitig Prune-Pakete, da jeweils der andere Router nicht auf dem Sendepfad zu *M1* liegt. Damit können sowohl *R3* als auch *R5* an ihre Vorgänger-Router ein Prune-Paket senden und in dessen Folge auch *R2*, da alle anderen Leitungen bereits abgeschnitten wurden. *R6* und *R7* hingegen leiten empfangene Prune-Pakete nicht an den Vorgänger weiter, da sich in ihrem LAN ein Multicast-Mitglied befindet. Diese haben zuvor mittels IGMP (siehe Abschnitt 2.3.3) ihren lokalen Router über die Gruppenzugehörigkeit informiert. Der entstandene Spannbaum ist in Abbildung 2.6 dargestellt.

Nach Ablauf einer Minute werden die Informationen über abgeschnittene Pfade auf den Routern verworfen und damit der belegte Speicherplatz freigegeben. Der nächste Sendevorgang führt somit wieder zu einem Fluten des Netzwerks mit anschließendem Beschneiden. Neben der Speicherfreigabe können dabei auch neue Mitglieder in den Spannbaum aufgenommen werden. Dies ist allerdings nur von sekundärer Bedeutung, da hierfür ein expliziter Mechanismus existiert. Sobald sich das erste Mitglied eines LANs mittels IGMP bei seinem lokalen Router zum Empfang einer Gruppe anmeldet, sendet dieser Router an seinen Vorgänger-Router ein Graft-Paket. Die notwendige Kenntnis des Vorgänger-Routers wird aus den zuvor gesendeten und gespeicherten Prune-Paketen ermittelt. Dies wird solange fortgeführt, bis ein Mitglied des Spannbaums erreicht wird. Der Empfang eines Graft-Pakets führt zur Wiedereingliederung des sendenden Routers in den Spannbaum. Abbildung 2.6b zeigt dies

beispielhaft für eine Anmeldung an Router *R3*. Wie daraus ersichtlich wird, antworten Router mit einem Bestätigungspaket auf empfangene Graft-Pakete. Bleibt dieses aus, so wird das Graft-Paket wiederholt gesendet.

Eine Zusammenfassung und Bewertung von DVMRP und der folgenden Routing-Protokolle erfolgt in Abschnitt 2.4.1.5.

#### 2.4.1.3 PIM-SM

Hinsichtlich der zweidimensionalen Einteilung der Routing-Protokolle aus Kapitel 2.4.1.1 ist Protocol-Independent-Multicast – Sparse-Mode (PIM-SM) [Estrin et al. 1998] das genaue Gegenteil von DVMRP. Anstatt für jeden Sender einen individuellen, optimierten Spannbaum aufzubauen, wird ein gemeinsamer von allen Sendern verwendet. Und anstatt das Netzwerk zu fluten, müssen Router generell explizit dem Spannbaum beitreten, um Multicast-Pakete zu erhalten. Dadurch bietet sich PIM-SM für verstreute Gruppen an. Die Bezeichnung „protocol-independent“ resultiert aus der Unabhängigkeit zum Unicast-Routing-Protokoll.

Der gemeinsame Spannbaum wird von einer zentralen Stelle aus aufgebaut, der sogenannten Rendezvous-Stelle. Die Rendezvous-Stelle wird von einem beliebigen PIM-Router gebildet. Sowohl die Sender als auch die Empfänger senden ihre Anmeldung in Richtung der Rendezvous-Stelle[4], um an den Spannbaum anzuknüpfen. Nachdem sich ein neuer Empfänger mittels IGMP bei seinem lokalen Router anmeldet, sendet dieser periodisch Anmeldungen zur Rendezvous-Stelle (siehe Abbildung 2.7a). Beim Weiterleiten der Anmeldungen legt dabei jeder beteiligte Router einen neuen Routing-Eintrag an, um Multicast-Pakete in die entgegengesetzte Richtung weiterzuleiten. Sobald auf einen Router gestoßen wird, der bereits Teil des Spannbaums ist (dies ist spätestens beim Erreichen der Rendezvous-Stelle der Fall), ist die Anmeldung abgeschlossen (siehe Abbildung 2.7b).

Ein Sender muss sich ebenfalls an der Rendezvous-Stelle anmelden. Dazu wird ein Anmeldepaket zusammen mit den ersten zu sendenden Daten per Unicast an die Rendezvous-Stelle gesendet. Die Rendezvous-Stelle antwortet mit einer Bestätigung der Anmeldung. Dabei wird wiederum bei allen beteiligten Routern die Routing-Tabelle ergänzt. Gleichzeitig wird das Datenpaket von der Rendezvous-Stelle per Multicast über den Spannbaum an alle Empfänger ausgeliefert (siehe Abbildung 2.8a). Alle weiteren Datenpakete aus demselben LAN können direkt mittels Multicast gesendet werden, anstatt sie in ein Unicast-Paket zu kapseln. Dennoch bedeutet dies nicht, dass sie direkt entlang des Spannbaums ausgeliefert werden, sondern dass sie wiederum bis zur Rendezvous-Stelle geleitet werden um dann von dort dem Spannbaum zu folgen. In Abbildung 2.8b ist dies dargestellt. Anstatt die

[4]Das Problem, die Rendezvous-Stelle aufzufinden, soll hier nicht vertieft werden. Den Routern ist die Rendezvous-Stelle bekannt. Dafür existiert ein Protokoll zum Austausch von Informationen zwischen den Routern und zur Wahl einer Rendezvous-Stelle [Estrin et al. 1998].

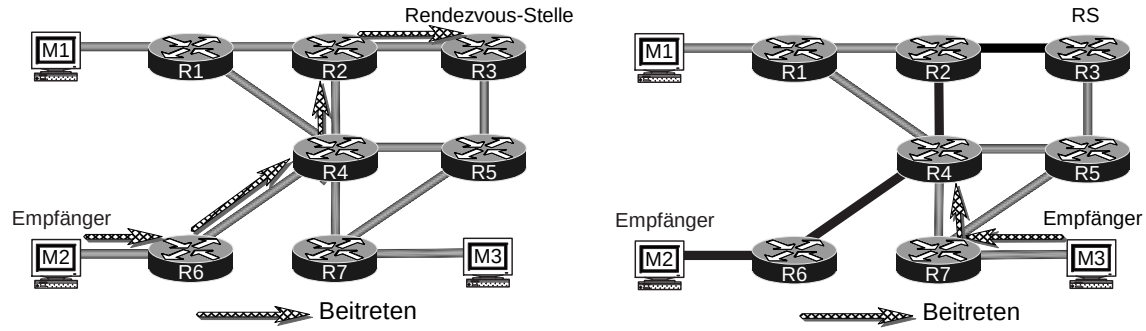



*Abbildung 2.7: PIM-SM: a) Beitritt des ersten Empfängers und b) Beitritt eines Empfängers bei bereits bestehendem Spannbaum*

Daten des Senders bereits bei Router *R2* über den Spannbaum auszuliefern, werden die Daten bis zur Rendezvous-Stelle *R3* weitergeleitet, um dann zurück an *R2* zu gehen und von dort weiter zu *M2* und *M3*. Der von PIM-SM etablierte Spannbaum wird daher auch unidirektional genannt, da er nur von einer Richtung aus, nämlich aus der Richtung der Rendezvous-Stelle, genutzt werden kann.

Der Abmeldevorgang ist dem Anmelden sehr ähnlich. Der lokale Router sendet ein Prune-Paket in Richtung der Rendezvous-Stelle. Dieses wird weitergeleitet und der Spannbaum dadurch entsprechend gekürzt bis ein Router erreicht wird, der noch weitere Multicast-Empfänger bedienen muss.

Eine Besonderheit von PIM-SM soll nicht unerwähnt bleiben. Wenn die Anzahl der Teilnehmer sehr groß wird, kann vom rendezvousbasierten Routing auf senderbasiertes Routing gewechselt werden. Dies führt zu optimierten Spannbäumen für jeden Empfänger, aber auch zu erhöhtem Verwaltungsaufwand bei den Routern. Der Wechsel zum senderbasierten Routing wird vom lokalen Router veranlasst. Jeder lokale Router entscheidet dies selbständig, d.h. zu einem Zeitpunkt kann sowohl sender– als auch rendezvousbasiertes Routing aktiviert sein. Um auf senderbasiertes Routing zu wechseln, sendet der lokale Router ein Join-Paket in Richtung des Multicast-Senders. Nachdem der Sender die Daten direkt empfängt, wird ein Prune-Paket in Richtung der Rendezvous-Stelle gesendet, damit die Multicast-Auslieferung über den gemeinsamen Spannbaum beendet wird. Wenn im weiteren Verlauf der Arbeit Bezug auf PIM-SM genommen wird dann ist immer rendezvousbasiertes Routing gemeint, da der Wechsel zu senderbasiertem Routing lediglich eine Option darstellt. Eine Diskussion über die Vor– und Nachteile von sender– und rendezvousbasiertem Routing wird in Abschnitt 2.4.1.5 geführt.

#### 2.4.1.4 Weitere Routing-Protokolle: CBT, MOSPF und PIM-DM

Die Routing-Protokolle Core-Based-Trees (CBT) [Ballardie 1997b, Ballardie 1997a], Multicast-Open-Shortest-Path-First (MOSPF) [Moy 1994] und Protocol-Independent-Multicast – Dense-Mode (PIM-DM) [Deering et al. 1997] sollen in diesem Kapitel kurz eingeführt werden. Die Erläuterungen fallen

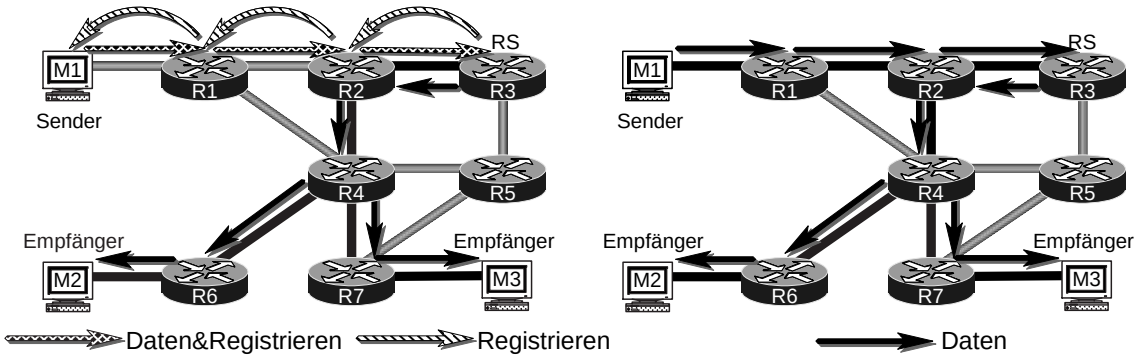



*Abbildung 2.8: PIM-SM: a) Registrierung eines Senders und b) Senden über den Spannbaum*

wesentlich knapper aus, da CBT und MOSPF nur in Einzelfällen eingesetzt werden und PIM-DM sehr ähnlich zu DVMRP ist.

CBT ist ein rendezvousbasiertes Verfahren, ähnlich dem von PIM-SM, d.h. es wird nur ein Spannbaum pro Gruppe aufgebaut. Eine Möglichkeit zum Wechseln auf senderbasierte Spannbäume ist nicht vorgesehen. Ein weiterer Unterschied zu PIM-SM stellen die bidirektionalen Spannbäume dar. Ein Paket muss nicht zuerst zur Rendezvous-Stelle gesendet und von dort aus verteilt werden, sondern kann direkt über den Spannbaum ausgeliefert werden. Im Vergleich mit Abbildung 2.8b zu PIM-SM würde das Paket vom Sender nur bis Router *R2* und von dort direkt an Router *R4* ausgeliefert werden.

MOSPF basiert auf dem Unicast-Routing-Protokoll OSPF. Wie in OSPF auch, werden in MOSPF aus Gründen der Skalierbarkeit autonome Systeme definiert und diese wiederum in Bereiche (Areas) aufgeteilt. Alle Bereiche sind disjunkt und über ein Backbone verbunden. Innerhalb eines Bereichs teilt jeder Router jedem anderen Router periodisch den Zustand seiner direkten Nachbarn und die Multicast-Gruppenzugehörigkeiten mit. Daraus kann ein Router für jeden Sender und jede Gruppe den minimalen Spannbaum nach dem kürzesten Pfad Algorithmus von Dijkstra berechnen [Dijkstra 1959]. Damit ist MOSPF wie DVMRP auch ein senderbasiertes Verfahren. Zwischen den Bereichen übernehmen Grenz-Router eine Stellvertreterfunktion für deren eigenen Bereich. Erzeugte Multicast-Pakete werden auch immer an den eigenen Grenz-Router vermittelt. Zwischen den Grenz-Routern werden wiederum Zustandspakete ausgetauscht und Spannbäume berechnet. Das Routing zwischen den autonomen Systemen ist nicht im Standard zu MOSPF definiert und kann daher prinzipiell beliebig sein. Eine einfache Möglichkeit ist RPF-Broadcasts zu versenden (siehe Abschnitt 2.4.1.2). MOSPF ist geeigneter für dichte Gruppen, da für verstreute Gruppen der Aufwand zum Austausch der Routing-Information zu aufwändig ist.

Abschließend soll PIM-DM noch kurz angesprochen werden. PIM-DM ist dem DVMRP-Verfahren sehr ähnlich. Im Gegensatz zu DVMRP ist es jedoch nicht auf RIP [Malkin 1994] als Unicast-Routing-Protokoll angewiesen. Stattdessen kann PIM-DM auf beliebige Unicast-Routing-Tabellen aufbauen.

#### 2.4.1.5 Bewertung der Routing-Protokolle

Für eine Bewertung der vorgestellten Routing-Protokolle sollen folgende Kriterien herangezogen werden: Senderentfernung im Spannbaum, Skalierbarkeit und schließlich praktische Verwendung. Die Senderentfernung wird in Teilstrecken (engl. hops) angegeben und ist die Entfernung jedes individuellen Gruppenmitglieds zum Sender der Gruppe. Ein bezüglich der Senderentfernung optimaler Spannbaum gliedert jeden Empfänger auf dem kürzesten Weg zum Sender in den Spannbaum ein, was eine geringe Paketverzögerung erzielt. Die Skalierbarkeit kann sowohl durch den entstehenden Berechnungsaufwand oder Speicheraufwand bei den Routern als auch durch den Kommunikationsaufwand im Netzwerk beschränkt sein.

DVMRP als Vertreter der senderbasierten Verfahren erzeugt für jeden Empfänger einen optimalen Spannbaum bezüglich der Entfernung (in Anzahl Routern) zum Sender, die minimal ist. Der Berechnungsaufwand bei den Routern ist gering. Dagegen ist der Speicheraufwand beträchtlich und es wird eine große Anzahl an Paketen versendet. Das Prinzip des Flutens und Zurückschneidens bedingt, dass alle Router für alle Paare <Sender, Gruppe> Statusinformation bezüglich des Vorgänger-Routers speichern auch wenn sie nicht zum Spannbaum gehören. Nur so ist eine nachfolgende Eingliederung mittels Graft-Paketen möglich, die an die Vorgänger-Router weitergeleitet werden müssen. Folglich skaliert DVMRP nicht mit der Anzahl der Multicast-Gruppen. DVMRP war lange Zeit das Multicast-Standardprotokoll und dürfte immer noch vorherrschend sein. Mit steigender Anzahl an Teilnehmern und Multicast-Routern wird die eingeschränkte Skalierbarkeit indessen zu einem ernsten Problem. Darum wird DVMRP zunehmend vor allem von PIM-SM verdrängt. Neben der Skalierbarkeit ist auch ein administratives Problem dafür verantwortlich. In der Vergangenheit kam es durch Fehler in den DVMRP-Implementierungen und fehlerhaft konfigurierter Software häufig vor, dass ein Router keine Prune-Pakete versendet hat. Dies führt zu einem Überfluten des angebundenen LANs mit Paketen zu allen aktiven Multicast-Gruppen und somit zu einem, sowohl für das LAN als auch für das Weitverkehrsnetz, ungünstigen Verhalten. Für PIM-DM gelten die hier gemachten Ausführungen ebenso. Jedoch gibt es vergleichsweise wenig Implementierungen im Gegensatz zu DVMRP und die praktische Bedeutung ist dementsprechend geringer.

PIM-SM bietet sowohl senderbasierte als auch rendezvousbasierte Spannbäume, infolgedessen kann zwischen optimalen Spannbäumen mit minimalen Entfernungen zwischen Sender und Empfänger oder einem möglichst niedrigem Aufwand im Netzwerk gewählt werden. Bei der Wahl von rendezvousbasiertem Routing werden vergleichsweise ungünstige Spannbäume erzeugt, da diese unidirektional sind und folglich ein Sender sein Paket prinzipiell zur Rendezvous-Stelle senden muss, um es von dort über den Spannbaum auszuliefern. Wenn senderbasierte und folglich optimale Spannbäume gewählt werden, muss das Netzwerk im Gegensatz zu DVMRP nicht geflutet werden. Nur Router, die Teil eines Spannbaums sind, halten Statusinformation. Die Skalierbarkeit kann aus diesem Grund als sehr gut eingestuft werden. Dies und die obige Ausführungen zu praktischen Problemen

bei DVMRP sind Hauptgrund für die wachsende Verbreitung von PIM-SM. Auch die Unabhängigkeit vom Unicast-Routing-Protokoll spricht dafür. Mehrere Nachteile von PIM-SM sollen demgegenüber nicht unerwähnt bleiben. So ist PIM-SM durch die Wahlmöglichkeit der Spannbaumart komplexer als DVMRP. Die Rendezvous-Stelle wirft das Problem auf wie diese gefunden wird. Deren Wahl hat großen Einfluss auf das Nachrichtenaufkommen und die Nachrichtenverzögerungen. Schließlich stellt die Rendezvous-Stelle auch einen single-point-of-failure bezüglich einer Multicast-Gruppe dar.

Einen weiterer Nachteil von PIM-SM, die unidirektionalen Spannbäume, behebt das CBT-Verfahren. Dafür kann nicht von rendezvous– auf senderbasiertes Routing gewechselt werden. Alle weiteren Kriterien entsprechen der Diskussion zu PIM-SM. Allerdings besitzt CBT kaum praktische Bedeutung.

MOSPF ist nicht nur in der Lage optimale Spannbäume aufzubauen, sondern erlaubt durch globale Information die sehr schnelle Anpassung auf Änderungen in der Gruppenmitgliedschaft oder in der Netzstruktur. Andererseits ist die Skalierbarkeit durch einen hohen Berechnungs– und Speicheraufwand bei den Routern stark eingeschränkt. MOSPF besitzt aus diesem Grund ebenfalls keine größere Bedeutung für die Praxis.

Im Verlauf dieser Arbeit werden die beiden Routing-Verfahren DVMRP und PIM-SM weiter von Bedeutung sein. DVMRP, weil es das gegenwärtig vorherrschende Routing-Verfahren ist und PIM-SM, weil es mit großer Wahrscheinlichkeit das zukünftige Standard-Routing-Verfahren wird. Eine Weiterentwicklung von PIM-SM, das senderspezifische PIM (PIM-SSM), wird zur Zeit überaus aktiv diskutiert. Dieses soll das Multicast-Routing vereinfachen, indem nur ein Sender pro Gruppe erlaubt ist. Eine Beschreibung findet sich in [Bhattacharyya et al. 2000a, Bhattacharyya et al. 2000b].

### 2.4.2 MBone — Das Multicast Backbone im Internet

Die Verwendung von Multicast bedingt eine erweiterte Funktionalität bei Routern und den daran angeschlossenen Rechnern, um einerseits das Gruppenmanagement mittels IGMP und andererseits das Weiterleiten der Pakete durch eines der angesprochenen Routing-Protokolle durchzuführen. Die Router des ursprünglichen, nur auf Unicast basierendem Internet, boten diese Funktionalität nicht an.

Anstatt nun alle Router in kürzester Zeit auswechseln zu müssen, wurde ein weitaus praktikablerer Weg beschritten, um einen Multicast-Dienst dennoch schnell verfügbar zu machen. Nach ersten Versuchen im DARTnet [Clark et al. 1991], einem Experimentalnetz finanziert von der DARPA, wurde 1992 das MBone (Multicast Backbone on the Internet) [Macedonia & Brutzman 1994, Svetz et al. 1996, Vinay 1996] gegründet. Das MBone ist ein virtuelles Netz, aufgesetzt auf das Internet, bestehend aus denjenigen Teilnetzen und Routern, die Multicast unterstützen. Zwischen den Multicast-Inseln wurden Tunnel eingerichtet, damit Multicast-Pakete das herkömmliche Unicast-Internet als Transitnetz verwenden können [Waitzman et al. 1988]. Das MBone soll die Zeit überbrücken, bis

das gesamte Internet Multicast unterstützt. Seit der Gründung wuchs das MBone mit exponentieller Geschwindigkeit und umfasste 1999 mehr als 12000 Teilnetze [Wittmann & Zitterbart 1999, S. 225].

Ursprünglich wurde im MBone ausschließlich DVMRP-Routing verwendet. Zu Beginn wurde DVMRP ohne Beschneidung des Spannbaums durch Prune-Pakete eingesetzt, was einem Broadcast über das gesamte MBone gleichkam. Mittlerweile wird der Prune-Mechanismus überall eingesetzt. Mehr und mehr wird allerdings dazu übergegangen PIM einzusetzen, welches mittlerweile ebenfalls in Standard-Routern implementiert ist.

## 2.5 Zuverlässiger Multicast auf Transportschicht

In den vorangegangenen Abschnitten wurden die Grundlagen von IP-Multicast eingeführt. Um einen zuverlässigen Multicast-Dienst anzubieten kann IP-Multicast verändert werden oder im Einklang mit dem ISO-OSI Referenzmodell IP-Multicast als Basisdienst von zuverlässigen Protokollen auf Transportschicht verwendet werden. In dieser Arbeit werden lediglich Transportschichtprotokolle betrachtet, die auf das Dienstmodell, die Adressierung und das Routing von IP-Multicast aufbauen. Die Darlegung der Beweggründe folgt in Abschnitt 2.5.2.3.

Nach einem einführenden Vergleich mit TCP wird eine Klassifikation zuverlässiger Multicast-Protokolle vorgestellt, die später in Kapitel 4 analysiert werden. Im weiteren Verlauf werden typische Vertreter jeder Klasse vorgestellt. Eine Auswahl der hier vorgestellten zuverlässigen Transportprotokolle wurde in einer Simulationsumgebung implementiert und miteinander verglichen. Die Ergebnisse werden ebenfalls in Kapitel 4 vorgestellt.

### 2.5.1 TCP und Fehlersemantik

Im Internet existiert mit dem Transmission-Control-Protocol (TCP) ein zuverlässiges Protokoll für die Unicast-Kommunikation. Ein erster, jedoch naiver Ansatz besteht darin, TCP auf Multicast-Kommunikation zu erweitern. Dies ist nicht praktikabel, da die vom Sender durchgeführte Fehlerdetektion sowie die Fluss– und Überlastkontrolle nicht auf eine Vielzahl von Empfängern direkt übertragbar ist. Schließlich ergibt sich ein Problem wenn man versucht die Semantik der zuverlässigen Übertragung zu definieren. Die Unicast-Semantik von TCP ist einfach zu definieren. An der Kommunikation ist ein Sender, ein Empfänger und das Netzwerk beteiligt. Ein temporärer Fehler beim Sender, im Netzwerk oder beim Empfänger wird durch Übertragungswiederholungen kompensiert. Die Semantik ist wie folgt: Arbeiten Sender, Empfänger und Netzwerk fehlerfrei oder mit korrigierbaren Fehlern, so werden schließlich alle Daten beim Empfänger ankommen. Bei einem nicht korrigierbaren Fehler beim Sender, Empfänger oder Netzwerk wird die Übertragung abgebrochen. Dies wird auch als Fate-Sharing-Prinzip in der Literatur bezeichnet [Clark 1988].

Beim Versuch dieses Fehlermodell auf einen Multicast-Dienst zu übertragen, stellt sich folgende Frage: was soll passieren, wenn nur ein Teil der Empfänger (oder das Netzwerk für einen Teil der Empfänger) einen nicht korrigierbaren Fehler aufweist, alle anderen Empfänger jedoch zuverlässig empfangen könnten? Soll die Übertragung abgebrochen werden (alles-oder-nichts Prinzip) oder soll versucht werden an möglichst viele Empfänger zuverlässig auszuliefern? Mittlerweile hat sich die Forschungsgemeinde darauf verständigt, dass es keine einzelne richtige Definition der Fehlersemantik für einen Multicast-Dienst geben kann, sondern dass die notwendige Fehlersemantik von der Anwendung abhängig ist [Mankin et al. 1998, Braudes & Zabele 1993, Diot et al. 1997]. Ansatzweise sind Diskussionen über die Fehlersemantik in [Floyd et al. 1997] und [Obraczka 1998] zu finden, meistens ergänzt durch weitere Dienstgütekriterien wie die Reihenfolgeeinhaltung. Eine umfassende Diskussion der möglichen Fehlersemantiken steht noch aus. Die Arbeit von Bäurle stellt einen ersten Versuch dar, diese Diskussion zu führen [Bäurle 1999]. Leider ist bei den veröffentlichten Transportprotokollen das Verhalten in den möglichen Fehlerfällen überwiegend nur sehr ungenau formuliert, sodass die Festlegung der Fehlersemantik für viele Protokolle nicht vollständig möglich ist. Im Rahmen der vorliegenden Arbeit wird diese Problemstellung nicht weiter vertieft. Im Folgenden wird bei der Betrachtung und Analyse der Mechanismen zur Sicherung der zuverlässigen Kommunikation, eine Beschränkung auf temporären Nachrichtenverlust im Netzwerk stattfinden. Dies stellt die häufigste Fehlerursache dar und beeinflusst aus diesem Grund hauptsächlich die Leistung der Protokolle.

### 2.5.2 Überblick über zuverlässige Multicast-Protokolle

Zuverlässige Kommunikation in Gegenwart von Nachrichtenverlust im Netzwerk versucht man durch zwei unterschiedliche Mechanismen, die auch kombiniert auftreten können, zu erreichen: Vorwärtsfehlerkorrektur und Quittierungen mit Übertragungswiederholungen. Bei der Vorwärtsfehlerkorrektur (engl. Forward Error Correction, FEC) werden, zusätzlich zu den Nutzdaten der Anwendung, Kontrolldaten übertragen [Rhee et al. 2000]. Diese ermöglichen, mit einer bestimmten Wahrscheinlichkeit, die Erkennung und Behebung von Fehlern. Ein garantiert zuverlässiger Dienst ist damit allerdings nicht zu erreichen. Eine garantierte Zuverlässigkeit wird mittels Quittierungsnachrichten und Übertragungswiederholungen realisiert. Dazu teilen die Empfänger dem Sender mit, welche Datennachrichten korrekt empfangen wurden (positive Quittungen) bzw. welche Datennachrichten nicht korrekt empfangen wurden (negative Quittungen). Der Sender initiiert dann gegebenenfalls Übertragungswiederholungen. Im Folgenden wird eine Unterscheidung in flache, hierarchische und Routerbasierte Verfahren getroffen.

#### 2.5.2.1 Flache Verfahren

Flache Verfahren zur Realisierung zuverlässiger Multicast-Kommunikation sind dadurch charakterisiert, dass die Quittungen direkt an den Sender gesendet werden. Handelt es sich dabei um positive Quittungen, so spricht man auch von senderinitiierten Verfahren, da der Sender dafür verantwortlich ist zu erkennen, ob ein Datenverlust aufgetreten ist. Finden negative Quittungen Verwendung, so ist jeder Empfänger für die Fehlerentdeckung verantwortlich. Daher klassifiziert man diese Verfahren auch als empfängerinitiiert. Beide Verfahren besitzen den Nachteil, dass der Sender durch die Verarbeitung zu vieler Quittierungen nicht beliebig viele Empfänger unterstützen kann, was zu hierarchischen Verfahren führt.

#### 2.5.2.2 Hierarchische Verfahren

Hierarchische Verfahren wurden entwickelt, um die Skalierbarkeit für beliebige Gruppengrößen zu ermöglichen. Die Quittungsimplosion beim Sender wird vermieden, indem alle Gruppenmitglieder in einem logischen *Kontrollbaum* mit dem Sender als Wurzel organisiert werden. Der Kontrollbaum kann unabhängig vom Spannbaum der Vermittlungsschicht sein. Quittierungen werden nicht mehr direkt an den Sender gesendet, sondern an den *Vaterknoten*[5] im Kontrollbaum. Der Vaterknoten eines Knotens ist der nächste Knoten im Kontrollbaum in Richtung der Wurzel. Diese werden im weiteren Verlauf dieser Arbeit als *Gruppenführer* bezeichnet. Ein Gruppenführer ist für das Senden von Übertragungswiederholungen an seine direkten Kindknoten zuständig. Ein Gruppenführer und seine Kindknoten werden als *lokale Gruppe* bezeichnet, im Gegensatz zur *globalen Gruppe*, die alle Empfänger einer Gruppe umfasst.

#### 2.5.2.3 Router-basierte Verfahren

Eine Reihe von Protokollen basieren auf Router-Unterstützung, die diesen Protokollen Vorteile gegenüber anderen Verfahren ermöglichen können, aber in gebräuchlichen Routern im Internet nicht zur Verfügung stehen. Diese Protokolle werden als Router-basierte Verfahren bezeichnet. Ihnen ist gemeinsam, dass die Router eine aktive Rolle für die zuverlässige Auslieferung übernehmen. Eine solche Router-Funktion ist das Auffinden eines geeigneten Gruppenführers, der im Auslieferungsbaum näher zum Sender liegt als die Empfänger seiner lokalen Gruppe. Damit wird erreicht, dass der Gruppenführer nicht die gleichen Nachrichten wie die lokale Gruppe verloren hat, sondern vermutlich weniger und daher selbständig Übertragungswiederholungen durchführen kann. Eine weitere Entlastung des Senders ist die Folge. Router-basierte Verfahren können aber auch so weit gehen, dass die

[5] Der zweisilbige Begriff *Knoten* wird als Alternative für den vor allem in zusammengesetzten Wörtern sperrigen dreisilbigen Begriff *Endsystem* verwendet, beinhaltet folglich die ISO-OSI-Schichten eins bis sieben.

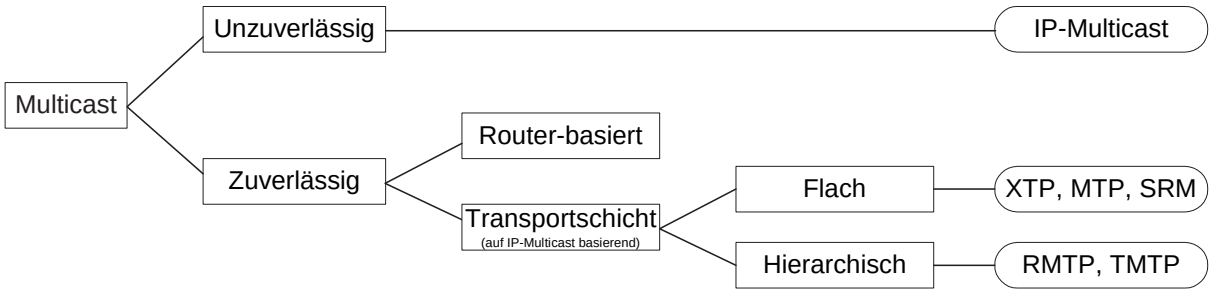



*Abbildung 2.9: Klassifikation von Multicast-Protokollen*

Router selbst Multicast-Nachrichten speichern und Übertragungswiederholungen durchführen [Lehman et al. 1998].

Eine Überlegung verdeutlicht, dass die Skalierbarkeit von Router-basierten Verfahren nicht ohne weiteres gegeben ist. Angenommen, ein Router ist Mitglied von 5000 Multicast-Gruppen. Dies ist eine Größe, die für zentral platzierte Router, beispielsweise als Knotenpunkte zwischen den Kontinenten, durchaus erreicht werden könnte. Für jede Gruppe muss der Router Datenpakete speichern, um Übertragungswiederholungen durchführen zu können. Angenommen, er muss durchschnittlich 30 Datenpakete zwischenspeichern und jedes Datenpaket umfasst 1,5 KB, dann speichert der Router alleine für die zuverlässige Multicast-Übertragung ungefähr 220 MB an Daten. Für nicht Router-basierte Verfahren ist es unwahrscheinlich, dass ein Empfänger an einer größeren Anzahl Multicast-Gruppen teilnimmt, was einen deutlich geringeren Speicherverbrauch verursacht. Wenn man nun berücksichtigt, dass diese Datenmenge effizient organisiert werden muss, dass Übertragungswiederholungen durchgeführt werden müssen und dass der Router daneben auch noch seine eigentliche Arbeit, das Vermitteln von Paketen erfüllen muss, dann wird deutlich, warum gegenwärtig derartige Verfahren nicht skalierbar eingesetzt werden können. Eine weitergehende Diskussion ist auch in [Hofmann 1998] zu finden.

Da Router-basierte Verfahren im Internet nicht ohne Änderungen der Router eingesetzt werden können, werden diese im Rahmen der vorliegenden Arbeit nicht weiter vertieft.

### 2.5.3 Protokollbeispiele

Im Folgenden werden beispielhaft drei flache, XTP, MTP und SRM, sowie zwei hierarchische zuverlässige Multicast-Protokolle, RMTP und TMTP, vorgestellt. Die bisherige Klassifikation ist in Abbildung 2.9 zusammengefasst. In Abschnitt 4.2 wird diese weiter detailliert und tiefer auf die verschiedenen Protokolle eingegangen. Schließlich werden die Simulationen in Abschnitt 4.7 stellvertretend drei der hier vorgestellten fünf Protokolle untersuchen.

#### 2.5.3.1 XTP

Das Xpress-Transport-Protocol (XTP) [Strayer et al. 1992] gehört zur Gruppe der flachen Verfahren. XTP bietet einen Multicast-Dienst im engeren Sinne an, d.h. keinen Multipeer-Dienst. Eine Besonderheit von XTP ist, dass die Empfänger ihre Quittungen nicht unaufgefordert an den Sender übermitteln, sondern dass der Sender die Empfänger dazu auffordert. Dies geschieht durch ein spezielles Flag im Kopf jeder gesendeten Dateneinheit. Die Quittierungen sind eine Mischung aus positiven und negativen Bestätigungen. Ein Empfänger sendet dazu eine Liste mit Sequenznummerbereichen die korrekt empfangen wurden an den Sender. Notwendige Übertragungswiederholungen werden vom Sender per Multicast an die gesamte Gruppe gesendet.

Neben der Zuverlässigkeit bietet XTP auch Mechanismen zur Flusskontrolle und Verbindungskontrolle an. Die Flusskontrolle basiert auf einem raten– und fensterbasierten Verfahren. Für den Sendekredit melden alle Empfänger ihre Wünsche an den Sender, der schließlich den minimalen Kredit, d.h. den schwächsten Empfänger, auswählt. Weiterhin stehen in XTP eine Vielzahl von Mechanismen zur Verfügung, mit denen ein Anwender das Protokoll parametrisieren kann. Dem steht allerdings eine nicht ausreichende Skalierbarkeit für große Gruppen gegenüber, da die Menge der zu verarbeitenden Quittierungen beim Sender von der Gruppengröße abhängig ist.

#### 2.5.3.2 MTP

Das Multicast-Transport-Protocol (MTP) [Armstrong et al. 1992] realisiert Multipeer-Kommunikation mit globaler Reihenfolgeerhaltung (Totalordnung hinsichtlich aller Sender). MTP gehört ebenfalls zu der Gruppe der flachen Verfahren und bietet darum nur begrenzte Skalierbarkeit. Der Nachrichtenverlust in MTP wird vom Empfänger erkannt, indem eine Nachricht mit höherer Sequenznummer als die erwartete empfangen wurde. Der Empfänger fordert die Wiederholung einer verlorenen Dateneinheit durch eine negative Quittung an, die per Unicast an den Sender übermittelt wird. Korrekt empfangene Daten werden dem Sender nicht bestätigt. Dieser überträgt eine angeforderte Nachrichtenwiederholung mittels Multicast an die gesamte Gruppe.

Bei einer genauerer Betrachtung dieser Mechanismen ergeben sich zwei Probleme. Die empfängerinitiierte Fehlererkennung erkennt Nachrichtenverlust erst beim korrekten Empfang der ersten darauffolgenden Datennachricht. Dies führt zu Problemen bei der Verlusterkennung der letzten zu sendenden Datennachricht und auch bei Datenübertragungen mit geringer Senderate. In MTP wird dies durch periodische Kontrolldateneinheiten behoben (engl. heartbeats), die gesendet werden, wenn innerhalb einer definierten Zeitspanne keine Nutzdaten übertragen werden. Ein weiteres Problem ist, dass der Sender nicht feststellen kann, ob alle Nachrichten erfolgreich ausgeliefert wurden (Nichtdeterminismus). Dies führt zu Problemen, wenn der Sender nicht alle gesendeten Daten beliebig lange für mögliche Übertragungswiederholungen speichern kann. In MTP garantiert der Sender für eine

wählbare Zeit das Senden von Übertragungswiederholungen. Negative Quittungen, die danach empfangen werden, führen zu keiner Übertragungswiederholung.

Neben der zuverlässigen Auslieferung bietet auch MTP eine raten– und fensterbasierte Flusskontrolle an. Die Ordnungserhaltung wird durch eine Sequentialisierung der Sendevorgänge über eine Master-Station erreicht. Diese sorgt dafür, dass alle Daten eine global eindeutige Nummerierung aufweisen.

#### 2.5.3.3 SRM

Scalable-Reliable-Multicast (SRM) [Floyd et al. 1997] ist ebenfalls der Klasse der flachen Verfahren zuzuordnen. Jedoch wird durch eine interessante Modifikation des Quittungsmechanismus eine weitaus bessere Skalierbarkeit als bei vergleichbaren Verfahren wie XTP oder MTP versprochen. Dies basiert auf der Erkenntnis, dass bereits eine einzige negative Quittung ausreicht, um für potentiell alle Gruppenmitglieder eine Übertragungswiederholung auszulösen. Wird der Datenverlust bei einem Empfänger bemerkt, so sendet dieser eine negative Quittung mittels Multicast an die gesamte Gruppe. Ebenfalls von diesem Nachrichtenverlust betroffene weitere Empfänger unterdrücken nun ihre eigene negative Quittung. Damit nicht zu viele Empfänger gleichzeitig eine negative Quittung senden, wird das Senden um eine zufällige Zeit verzögert [Ramakrishnan & Jain 1987]. Prinzipiell können außer dem Sender beliebige andere Empfänger, welche die nachgeforderte Dateneinheit korrekt empfangen haben, die Übertragungswiederholung durchführen, was die Skalierbarkeit weiter verbessern kann.

Demgegenüber besitzt dieses Verfahrens auch einen bedeutsamen Nachteil. Ein Empfänger mit einer hohen Fehlerrate kann durch seine permanenten, an die gesamte Gruppe gesendeten negativen Quittungen, alle anderen Teilnehmer empfindlich stören, was die Skalierbarkeit beeinträchtigt. Dieses Problem wird auch als „crying baby“-Problem bezeichnet [Holbrook et al. 1995, Levine & Garcia-Luna-Aceves 1996].

Dem bei MTP angesprochenen prinzipiellen Problem der empfängerinitiierten Fehlererkennung, die erst durch eine weitere erfolgreich empfangene Datennachricht erfolgt, wird durch regelmäßige Statusmeldungen aller Gruppenmitglieder an die Gruppe begegnet. Die Statusmeldungen enthalten unter anderem die höchste korrekt empfangene Sequenznummer. Durch Steuerung der Intervallzeit zwischen zwei Statusmeldungen in Abhängigkeit von der Gruppengröße wird versucht, das Verkehrsaufkommen in einem akzeptablen Rahmen zu halten. Dennoch können derartige positive Quittungen wiederum ein Problem für große Gruppen darstellen.

Um die Beschreibung von SRM abzurunden, sei noch darauf hingewiesen, dass es für eine Whiteboard-Anwendung [Jacobson 1992] im MBone entwickelt wurde. Eine weitere interessante Unterscheidung zu gängigen anderen Verfahren ergibt sich durch die starke Orientierung an dem ALF-Konzept (Application Level Framing) [Clark & Tennenhouse 1990]. Diesem Ansatz liegt die Annahme zugrunde, dass die verschiedenartigen Anforderungen der Applikationen nicht durch einen festen

Satz von Protokollfunktionen adäquat unterstützt werden können. Folglich sollten die Protokolle nur Grundfunktionen zur Verfügung stellen, die von den Anwendungen erweitert werden. Entsprechend diesem Konzept muss beispielsweise die Ordnungserhaltung oder Flusskontrolle von der Anwendung und nicht vom Transportprotokoll erbracht werden.

#### 2.5.3.4 RMTP

Mit dem Reliable-Multicast-Transport-Protokoll (RMTP) [Lin & Paul 1996, Paul et al. 1997] wird das erste hierarchische Protokoll vorgestellt. Die Grundzüge der Konzepte wurden bereits in [Paul et al. 1994] beschrieben. Alle Empfänger sind Teil des Kontrollbaums mit dem Sender an der Wurzel. Teilmengen der Empfänger werden zu lokalen Gruppen zusammengefasst und einem Gruppenführer zugeordnet. Gruppenführer und normale Empfänger werden wiederum einem übergeordneten Gruppenführer zugeordnet, bis ein Kontrollbaum wie in Abbildung 2.10 entsteht. Gruppenführer sind dabei normale Empfänger, die zusätzlich die Aufgabe erhalten, die Quittungen ihrer lokalen Gruppe zu bearbeiten und gegebenenfalls Übertragungswiederholungen zu senden. Dazu senden die Empfänger ihre positiven Quittungen an die zugeordneten Gruppenführer. Die Spezifikation von RMTP legt fest, dass die Quittungen nicht nach jeder Datenübertragung, sondern periodisch gesendet werden. Der Gruppenführer verhält sich in Richtung der Wurzel des Kontrollbaums wie ein normaler Empfänger, d.h. er sendet ebenfalls periodisch positive Quittungen zur Bestätigung der empfangenen Daten.

Übertragungswiederholungen werden, abhängig von der Anzahl der Nachforderungen, entweder mittels Unicast oder Multicast durchgeführt. Wird die Übertragungswiederholung mit Multicast durchgeführt, so möchte man erreichen, dass diese nur an den Teilbaum ausgeliefert wird, der vom Verlust betroffen ist. Das folgende Beispiel soll dies verdeutlichen. Angenommen, in Abbildung 2.10 möchten $E_1$ und $E_2$ eine Übertragungswiederholung von $D_1$ und diese soll mit Multicast erfolgen. Diese Wiederholung sollte idealerweise nicht an alle Empfänger gesendet werden, sondern nur an den Teilbaum von Router $R_2$. Dazu führt RMTP eine Technik mit dem Namen „Subcasting“ ein. Beim Subcasting soll eine Nachricht nur an den eigenen Teilbaum (vom Sender als Wurzel aus gesehen) gesendet werden. In IP-Multicast ist eine derartige Funktion nicht vorgesehen, deswegen werden in RMTP modifizierte Router verwendet, die das Subcasting beherrschen. Strenggenommen müsste RMTP folglich den Router-basierten Verfahren zugeordnet werden. Dadurch ist ein Einsatz im gegenwärtigen Internet nicht möglich. Auch die Annahme, dass die Gruppenführer aufgrund existierenden Wissens über die Topologie des Netzwerks statisch ausgewählt werden, erschwert den realen Einsatz. Eine Anpassung auf die von IP-Multicast bereitgestellten Mechanismen ist allerdings einfach möglich und wird auch praktiziert, weshalb RMTP hier nicht zu den Router-basierten Verfahren gezählt wird.

Neben der Fehlerkontrolle definiert RMTP auch Mechanismen zur Fluss– und Überlastkontrolle. Für die Flusskontrolle wird ein raten– und fensterbasiertes Verfahren verwendet. Dazu enthalten die beim

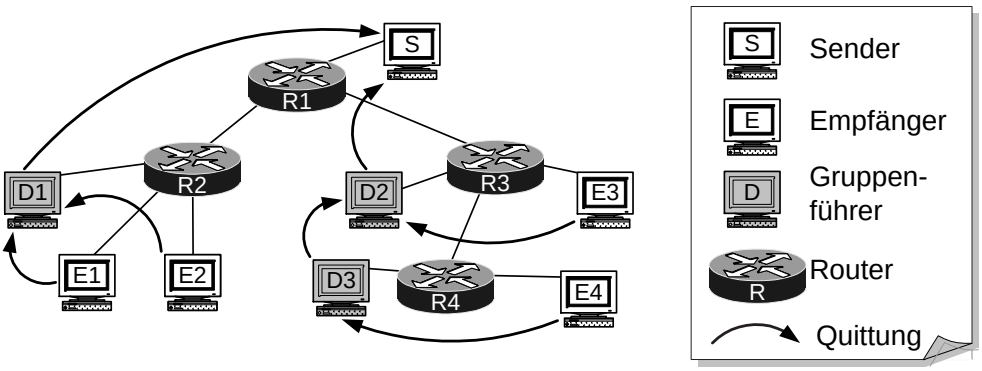



*Abbildung 2.10: Kontrollbaum bei RMTP*

Sender eingehenden positiven Quittungen zusätzlich die Größe des Empfängerfensters. Ausschlaggebend für die Erhöhung des Sendekredits ist der langsamste Empfänger. Für die Überlastkontrolle werden ebenfalls die Quittungen der Empfänger ausgewertet. Übersteigen die Nachforderungen eine Grenze, so wird von einer Überlastung des Netzwerks ausgegangen und der Sendekredit verringert. Der Mechanismus ist ähnlich dem Slow-Start-Mechanismus von TCP [Jacobson & Karels 1988].

#### 2.5.3.5 TMTP

Auch das Tree-based-Multicast-Transport-Protocol (TMTP) [Yavatkar et al. 1995] verwendet eine hierarchische Ordnung aller Empfänger, um Skalierbarkeit auch für große Gruppen zu gewährleisten. Ein bedeutender Unterschied ergibt sich durch die Verwendung von negativen Quittungen mit Quittungsunterdrückung ähnlich zu SRM (Abschnitt 2.5.3.3). Durch die Verwendung von negativen Quittungen mit Quittungsunterdrückung müssen Übertragungswiederholungen prinzipiell per Multicast gesendet werden. Auch hier wird die Forderung gestellt, Übertragungswiederholungen möglichst nur an die Mitglieder einer lokalen Gruppe zu versenden. TMTP verwendet dazu einen mit dem Ausbreitungsbereich (engl. time-to-live, TTL) von IP-Paketen gesteuerten Mechanismus [Postel 1981b]. Der Gruppenführer sendet Übertragungswiederholungen mit einem TTL-Wert, der auf den entferntesten Empfänger der lokalen Gruppe angepasst ist. Weiter entfernte Empfänger erhalten diese Pakete nicht. Für den Baumaufbau wird die erweiternde Ringsuche (expanding ring search, ERS) [Boggs 1983] verwendet, die später noch genauer erläutert wird.

Für die Flusskontrolle sieht TMTP ebenfalls eine Kombination aus raten– und fensterbasiertem Verfahren vor. Zum Weiterschalten des Sendefensters werden zusätzlich zu den negativen Quittungen auch periodisch positive Quittungen an den Gruppenführer gesendet. Neben der Notwendigkeit für die Flusskontrolle wird dadurch auch das Problem des Nichtdeterminismus, wie in Abschnitt 2.5.3.2 diskutiert, gelöst.

## 2.6 Zusammenfassung und Diskussion

Mit der Beschreibung von IGMP und den verschiedenen Routing-Verfahren wurde die Grundlage für das Verständnis der nachfolgenden Kapitel gelegt. Vor allem die verschiedenen Routing-Verfahren mit ihren speziellen Eigenschaften müssen beim Entwurf von Multicast-Protokollen oder Diensten für Multicast-Protokolle berücksichtigt werden. In Kapitel 3.2 wird dies noch ausführlicher erläutert.

In dem gegebenen Überblick über zuverlässige Multicast-Dienste konnte ein Ausschnitt von Protokollen behandelt werden, deren Verfahren oft stellvertretend für viele andere Protokolle stehen. Es wird davon ausgegangen, dass eine Verbesserung der Skalierbarkeit durch negative Quittungen, eine weitere Verbesserung durch negative Quittungen mit Quittungsunterdrückung und schließlich die beste Skalierbarkeit durch hierarchische Verfahren erreicht wird. In der Literatur gibt es dafür zahlreiche Argumente, aufbauend auf Analysen oder Simulationen [Levine & Garcia-Luna-Aceves 1998, Hashimoto et al. 1999].

Für hierarchische Protokolle stellt sich die entscheidende Frage wie die Kontrollbäume zu erstellen sind. TMTP verwendet dazu die erweiternde Ringsuche (ERS). In Kapitel 3 werden weitere Verfahren vorgestellt. Daran schließt sich ein umfassender Vergleich dieser Verfahren, sowohl argumentativ als auch durch Simulationen, an. Vorwegnehmend sei gesagt, allen bisherigen Verfahren ist gemein, dass sie keine ausreichende Skalierbarkeit bieten und aufgrund weiterer spezifischer Nachteile keinen universellen Einsatz erlauben. Dieser unbefriedigenden Situation soll durch ein neues Verfahren zum Aufbau von Kontrollbäumen, dem sogenannten Token-Repository-Service, Rechnung getragen werden. Der ausführlichen Beschreibung folgen Analysen und Simulationen, die die vorteilhaften Eigenschaften des Token-Repository-Service belegen. Es zeigt sich, dass erst mit dem Token-Repository-Service hierarchische Transportprotokolle für große Teilnehmerzahlen eingesetzt werden können.

In Kapitel 4 werden wichtige Klassen zuverlässiger Transportprotokolle einer detaillierten Analyse unterzogen. Bisherige Arbeiten gingen von einfachen Modellen aus oder untersuchten keine allgemeinen Protokollklassen, sondern zeigten, dass ein spezifisches Protokoll in bestimmten Situationen Vorteile besitzt. Viele interessante und für eine Protokollauswahl zentrale Fragen sind noch ungeklärt, so beispielsweise der Bandbreitenverbrauch bei Sender und Empfängern oder wie sich hierarchische Protokolle auf die Nachrichtenverzögerung auswirken. Neben der Bestätigung, dass lediglich hierarchische Protokolle ausreichende Skalierbarkeit für große Teilnehmerzahlen bieten, werden weitere Vorteile dieser Protokollklasse, wie geringerer Bandbreitenverbrauch und geringere Nachrichtenverzögerung, aufgezeigt. Schließlich wird sich zeigen, dass mit dem Token-Repository-Service die Vorteile hierarchischer Verfahren ausgeweitet werden können, da gewünschte Eigenschaften des entstehenden Kontrollbaums zur Leistungsoptimierung flexibel konfiguriert werden können.

# Kapitel 3

# Kontrollbaumaufbau für hierarchische Multicast-Transportprotokolle

## 3.1 Einführung

Für hierarchische Transportprotokolle ergibt sich der entscheidende Vorteil aus der Skalierbarkeit, die auch für sehr große Empfängergruppen gegeben ist. Dies wird erreicht, indem die Last für Quittungsbearbeitung und Übertragungswiederholungen im Kontrollbaum auf die lokalen Gruppen verteilt wird, anstatt nur auf den Sender konzentriert zu sein. Da die Größe lokaler Gruppen begrenzt ist, muss jeder Gruppenführer unabhängig von der Teilnehmerzahl der Multicast-Gruppe, lediglich für eine begrenzte Empfängeranzahl Quittungen sammeln und Übertragungswiederholungen durchführen. Entsprechend trifft dies für den Sender zu, der ebenfalls ein Gruppenführer ist. Mit dieser Maßnahme wird die Überlastung einzelner Rechnerknoten wirksam vermieden und Skalierbarkeit für beliebige Teilnehmerzahlen erreicht. In Kapitel 4 wird dies belegt werden.

Einem Einsatz hierarchischer Transportprotokolle geht der notwendige Aufbau des Kontrollbaums voraus. Die zu lösende Aufgabe besteht darin, einen neuen Teilnehmer, welcher der zuverlässigen Multicast-Gruppe beitritt, in den Kontrollbaum zu integrieren, d.h. einen Gruppenführer zu finden, der den neuen Teilnehmer in seine lokale Gruppe aufnimmt. Das wesentliche Entwurfskriterium eines Verfahrens zum Kontrollbaumaufbau muss eine skalierbare und effiziente Realisierung sein, um große Gruppen– und Teilnehmerzahlen adäquat unterstützen zu können.

Im nächsten Abschnitt werden die bisher verwendeten Verfahren beschrieben, um einen Kontrollbaum zu etablieren. Anschließend wird mit dem Token-Repository-Service ein neuer Ansatz vorgestellt, der entscheidende Vorteile gegenüber bisherigen Verfahren hat. Dies wird zum einen durch eine Leistungsanalyse am Ende dieses Kapitels gezeigt, zum anderen aber auch durch die Simulationsergebnisse in Abschnitt 3.10 unterstrichen.

## 3.2 Verwandte Arbeiten

Die verwandten Arbeiten zum Kontrollbaumaufbau werden unterschieden in Transportschichtprotokolle, die keine Modifikationen der Router voraussetzen und Router-basierte Protokolle, die Modifikationen erfordern und deshalb mit üblichen Routern nicht zusammenarbeiten.

### 3.2.1 Transportschichtprotokolle

Die Grundlage für die meisten Ansätze zum Aufbau hierarchischer Kontrollbäume bildet die erweiternde Ringsuche (engl. expanding ring search, ERS). Diese wurde erstmals in [Boggs 1983] vorgestellt, damals allerdings noch nicht, um Kontrollbäume zu erstellen, sondern allgemein, um nach Ressourcen in einem Netzwerk zu suchen. Die Annahme ist, dass nach einer Ressource aus einer Menge von gleichen Ressourcen gesucht wird. Die gefundene Ressource soll möglichst „nahe“ am suchenden Knoten sein. In der erweiternden Ringsuche wird die Entfernung in Teilstrecken (engl. hops), d.h. in der Anzahl dazwischenliegender Router, ausgedrückt. Eine Voraussetzung für ERS ist die Vereinbarung zwischen dem Klienten (suchender Knoten) und dem Server (gesuchter Knoten oder Ressource), eine gemeinsame Multicast-Adresse zu verwenden. Der Klient sendet eine Nachricht an die bekannte Multicast-Adresse. Auf eine globale Anfrage würden möglicherweise sehr viele der gesuchten Server antworten und es wäre schwierig, den nächstliegenden zu ermitteln. Deshalb wird die erste Suchnachricht mit einer Lebenszeit (TTL) von eins versendet. Dadurch bleibt die Nachricht auf das lokale Netz beschränkt und wird von den Routern nicht weitergeleitet. Der Klient wartet nun eine angemessene Zeitspanne auf Antwort. Trifft während der Wartezeit keine Antwort ein, so wird die Suchnachricht wiederholt, diesmal allerdings mit erhöhtem TTL. Ein TTL von zwei legt fest, dass die Nachricht genau einmal von einem Router weitergeleitet wird (siehe Abbildung 3.1). Ebenso muss die Wartezeit nun verlängert werden, um die längeren Nachrichtenlaufzeiten zu berücksichtigen. Dieser Vorgang wird wiederholt bis schließlich eine Nachricht eintrifft oder der maximale TTL von 255 erreicht wird. Dies bedeutet, dass kein Server gefunden werden konnte.

Aus der Sichtweise eines Servers sieht die Situation folgendermaßen aus. Trifft eine Suchnachricht ein, so antwortet er mit einer Unicast-Nachricht an den Klienten. Für den Einsatz von ERS zum Kontrollbaumaufbau ist der Klient ein neuer Multicast-Teilnehmer, der in den Kontrollbaum integriert werden möchte. Die gesuchte Ressource sind die Knoten des Kontrollbaums. Es antworten indes nur die Knoten des Kontrollbaums, die ihren maximalen Verzweigungsgrad noch nicht erreicht haben. Der *Verzweigungsgrad* besagt, wie viele Kindknoten ein Gruppenführer durchschnittlich im Kontrollbaum besitzt. Der *maximale Verzweigungsgrad* ist dementsprechend die maximale Anzahl Kindknoten, die ein Gruppenführer akzeptieren kann und ein konfigurierbarer Parameter beim Kontrollbaumaufbau.

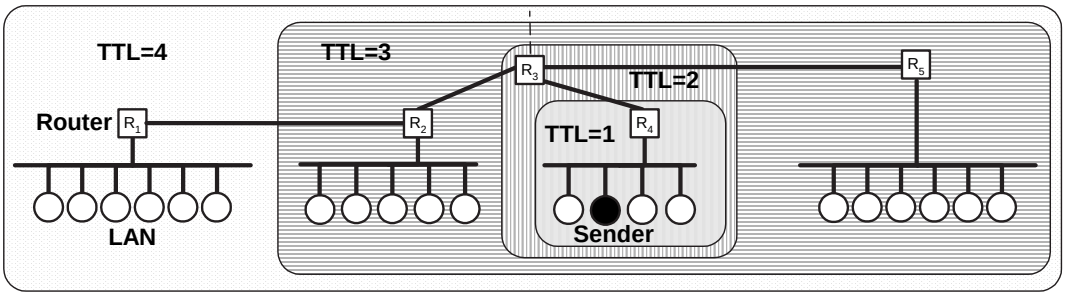



*Abbildung 3.1: Ausbreitungsbereich von Suchnachrichten*

Knoten, die ihren maximalen Verzweigungsgrad noch nicht erreicht haben, werden *unvollständig verzweigte Knoten* genannt. Nachdem der neue Teilnehmer eine Antwort von einem unvollständig verzweigten Knoten erhalten hat, kann er sich an den antwortenden Knoten als seinen Vaterknoten im Kontrollbaum anmelden.

Beispiele für Protokolle, die ERS verwenden, sind TMTP [Yavatkar et al. 1995], TRAM [Chiu et al. 1998] und Lorax [Levine et al. 1996]. TMTP und TRAM verwenden ERS in der beschriebenen Form, während Lorax den ERS-Algorithmus modifiziert. Um einen sehr großen Ausbreitungsbereich der Suchnachrichten zu verhindern, kann ein anderer Knoten, der ebenfalls nicht zum Kontrollbaum gehört, von dem aber angenommen wird, dass er näher zum Kontrollbaum platziert ist, seine Hilfe anbieten. Als Folge davon stellt der ursprüngliche Knoten das Senden von Suchnachrichten ein; stattdessen sendet fortan der Hilfsknoten. Lorax und TRAM verwenden neben ERS eine weitere Eingliederungstechnik, die ERA genannt werden soll und nachfolgend beschrieben wird.

Eine gängige Alternative zu ERS stellt das Vertauschen der Suchrichtung dar. Nicht neue Teilnehmer suchen nach Vaterknoten im Kontrollbaum, sondern Knoten des Kontrollbaums suchen nach neuen Teilnehmern. Dazu senden die unvollständig verzweigten Knoten periodisch Suchnachrichten mit Multicast auf eine von beiden Suchpartnern bekannte Multicast-Adresse. Auch dieses Aussenden kann mit erweiterndem TTL geschehen, um die Lokalität der Suche zu verbessern und gleichzeitig die Netzlast zu reduzieren. Das Verfahren mit umgekehrter Suchrichtung wird erweiternde Ringwerbung (engl. expanding ring advertisement, ERA) genannt. Die erweiternde Ringwerbung kann als Vorteil für sich verbuchen, dass zusammen mit den Suchnachrichten auch Sitzungsinformationen übertragen werden können, z.B. über bisher bereits gesendete Nachrichten und Nachrichten, die noch gespeichert sind und Übertragungswiederholungen erlauben. Eine Besonderheit ergibt sich aus dem periodischen und nicht ereignisgesteuerten Senden von Suchnachrichten. Dieses bedingt eine Abwägung zwischen der Netzlast und der Zeitverzögerung für den Gruppeneintritt durchzuführen, um die Periodenzeit festzulegen.

Neben Lorax und TRAM verwenden RMTP [Paul et al. 1997] und LGC [Hofmann 1998] das ERA-Verfahren. Das LGC-Verfahren modifiziert die Grundversion von ERA allerdings dahingehend, dass

der Ausbreitungsbereich und die Zeitintervalle der Suchnachrichten entsprechend einer Tabelle ermittelt werden. Dadurch wird erreicht, dass Nachrichten mit einem großen Ausbreitungsbereich seltener als lokale Nachrichten versendet werden. Dies reduziert zum einen die Netzlast durch weniger Weitverkehrsnachrichten, zum anderen empfängt ein Klient aber auch mehr Nachrichten von Vaterknoten in der Nähe, die natürlich zu bevorzugen sind.

### 3.2.2 Router-basierte Protokolle

Neben diesen beiden Ansätzen auf der Transportschicht bauen andere Verfahren auf die Unterstützung von Routern. Ein neueres Protokoll dafür ist Pragmatic-General-Multicast (PGM) [Speakman et al. 2000]. In PGM müssen sich neue Mitglieder nicht explizit anmelden. PGM verfolgt den Ansatz, keinen separaten Kontrollbaum auf Transportschicht aufzubauen, sondern den Multicast-Routing-Baum auch zur Fehlerkontrolle und Fehlerbehebung zu nutzen. Der Kontrollbaum besteht dabei aus PGM-fähigen Routern, die über ihren Vorgänger im Kontrollbaum informiert werden müssen, d.h. über den nächsten PGM-Router der näher zum Sender liegt. Dazu werden vom Sender Verwaltungsnachrichten an die Multicast-Gruppe periodisch versendet, die aber von den Routern nicht direkt weitergeleitet werden, sondern verarbeitet und modifiziert werden müssen. Dadurch erfährt ein Router die Identität seines Vorgängers im Kontrollbaum und teilt über Modifikation des Multicast-Pakets und dessen Weiterleitung dem nächsten Router die eigene Identität mit. Aufgrund der nun vorhandenen Vorgängerinformation ist ein Kontrollbaum aus Routern entstanden. In der Grundversion von PGM erlaubt dieser Kontrollbaum zwar die Aggregation und Unterdrückung unnötiger Quittungen, allerdings müssen alle Übertragungswiederholungen vom Sender durchgeführt werden, da die Router keine Datenpakete speichern.

Weitere Router-basierte Ansätze besitzen ähnliche Mechanismen wie PGM, um einen impliziten oder expliziten Kontrollbaum, bestehend aus Routern und eventuell weiteren Knoten, aufzubauen. Ein Konzept des Lightweight-Multicast-Service (LMS) [Papadopoulos et al. 1998] besteht darin, dass diejenigen Router Informationen über Knoten speichern und periodisch austauschen, die Nachrichtenwiederholungen durchführen können. Bei Routern eingehende Aufforderungen für Übertragungswiederholungen müssen dazu von den Routern interpretiert und an einen geeigneten Knoten weitergeleitet werden. Im Gegensatz dazu wird bei OTERS [Li & Cheriton 1998] der Kontrollbaum explizit, jedoch mit Unterstützung der Router, aufgebaut. Jedes Mitglied kennt demnach seinen Vorgängerknoten im Kontrollbaum und kann Übertragungswiederholungen direkt beim Vorgänger anfordern, ohne dabei auf die Mithilfe der Router angewiesen zu sein.

Während die bisher beschriebenen Router-basierten Verfahren zwar spezielle Funktionalitäten bei Routern voraussetzen, aber dennoch bestrebt sind, deren Last möglichst gering zu halten, geht Active-Reliable-Multicast (ARM) [Lehman et al. 1998] einen anderen Weg. Hier werden Multicast-Pakete im

Speicher des Routers abgelegt, damit diese selbst Übertragungswiederholungen durchführen können. Die Etablierung eines expliziten Kontrollbaums entfällt dadurch vollständig.

### 3.2.3 Diskussion und Abgrenzung

Einer Diskussion der vorhandenen Lösungsansätze für den Kontrollbaumaufbau soll eine Festlegung der notwendigen und wünschenswerten Ziele vorausgehen, an denen sich die Protokolle messen müssen. Offensichtlich ist, dass die Protokolle selbst zuverlässig und fehlertolerant arbeiten sollen. Diese Anforderung ist bei allen Lösungsansätzen gegeben. Die folgenden Kriterien sollen für eine Beurteilung herangezogen werden:

1. Universelle Einsetzbarkeit: In vielen Protokollbeschreibungen in der Literatur wird der universellen Einsetzbarkeit zu wenig Aufmerksamkeit geschenkt. Manche Ansätze treffen teils implizite Annahmen über die Struktur und Funktionalität eines Netzwerks, die nicht überall und vor allem nicht weiträumig zutreffen und die universelle Verwendbarkeit stark einschränken. Ein Beispiel für eine derartige Annahme sind bidirektionale Multicast-Netze.

2. Skalierbarkeit: Der Kontrollbaumaufbau führt zu zusätzlicher Netz– und Knotenlast durch den notwendigen Nachrichtenaustausch. Jene zusätzliche Last soll möglichst gering sein und zudem auch für große Teilnehmerzahlen skalieren, um die gewonnene Skalierbarkeit von hierarchischen Transportprotokollen nicht durch Engpässe beim Aufbau des Kontrollbaums wieder zu verlieren.

3. Zeitbedarf für den Gruppeneintritt: Vor allem für sehr dynamische Gruppen ist der Zeitbedarf für den Gruppeneintritt, der möglichst gering sein sollte, ein wichtiges Kriterium. Durch einen schnellen Gruppeneintritt wird zudem sichergestellt, dass ein neues Mitglied rasch als Gruppenführer für weitere eintretende Teilnehmer in der näheren Netzwerkumgebung zur Verfügung steht und sich somit kurze Kommunikationswege im Kontrollbaum ergeben (siehe dazu auch die Simulationsergebnisse in Abschnitt 3.10.2.1).

4. Kontrollbaum am Routing-Baum orientiert: Um möglichst kurze Wege innerhalb des Kontrollbaums und eine lokale Fehlerbehebung zu verwirklichen, soll der Kontrollbaum die Struktur des Routing-Baums widerspiegeln. Dazu gehören die folgenden beiden Eigenschaften: erstens sollen Vater– und Kindknoten des Kontrollbaums auch im Routing-Baum bezüglich der Teilstreckenanzahl nahe beieinander liegen, um kurze Kommunikationswege und schnelle Fehlerbehebung zu ermöglichen. Und zweitens soll ein Vaterknoten im Kontrollbaum auch im Routing-Baum bezüglich der Teilstreckenanzahl näher am Sender liegen als seine Kindknoten (siehe auch Abschnitt 3.7.2). Wird diese Forderung eingehalten, dann steigt die Wahrscheinlichkeit

für erfolgreichen Nachrichtenempfang mit abnehmender Entfernung zur Wurzel im Kontrollbaum. Dies ist die Voraussetzung dafür, dass Gruppenführer mit hoher Wahrscheinlichkeit eine lokale Übertragungswiederholung durchführen können und die Kontrollbaumwurzel entlastet wird.

5. Zuverlässigkeit des Kontrollbaums: Für die Zuverlässigkeit des Kontrollbaums ist es vorteilhaft, wenn beim Gruppenaustritt eines Teilnehmers oder dessen Ausfall möglichst wenig Kindknoten betroffen sind und folglich der Reorganisationsaufwand möglichst gering ausfällt. Dies wird erreicht, indem die Höhe des Kontrollbaums minimiert wird (siehe Abschnitt 3.5.2 für eine genaue Definition).

Die universelle Einsetzbarkeit von ERS ist aus mehreren Gründen nicht gegeben. Eine wichtige implizite Voraussetzung von ERS ist ein symmetrisches und bidirektionales Multicast-Netzwerk. So kann ERS nicht angewendet werden, wenn das Netzwerk keinen echten Multicast-Dienst auf der Vermittlungsschicht bereitstellt. Eine weitere Einschränkung betrifft unidirektionale Multicast-Netzwerke, z.B. die wichtige Klasse der Satellitennetzwerke. Diese besitzen nur einen unidirektionalen Multicast-Kanal und keinen oder nur einen überaus ineffizienten Multicast-Rückkanal. Selbst wenn ein Rückkanal existiert, so ist dieser nicht symmetrisch zum Hinkanal, was in der Etablierung eines ungünstigen Kontrollbaums resultieren kann, der nicht am Routing-Baum orientiert ist.

Ein weiteres grundlegendes Problem, das die universelle Verwendbarkeit von ERS beeinträchtigt, ist die starke Abhängigkeit vom Multicast-Routing-Protokoll. Gegenwärtig wird intensiv über senderspezifisches PIM [Bhattacharyya et al. 2000a, Bhattacharyya et al. 2000b] als Standard für Routing-Verfahren diskutiert (siehe Abschnitt 2.4.1.5). Dieses erlaubt lediglich einen Sender pro Multicast-Gruppe, vereinfacht dafür aber das Routing erheblich. Da die Chancen für senderspezifisches PIM recht gut stehen ergibt sich für den Einsatz von ERS ein Problem. ERS ist auf beliebig viele Sender pro Multicast-Gruppe angewiesen und kann nicht mit senderspezifischem PIM zusammenarbeiten. Doch auch von den anderen Routing-Verfahren arbeitet ERS nur mit DVMRP oder ähnlichen senderbasierten Routing-Protokollen sinnvoll zusammen, was die Orientierung des Kontrollbaums am Routing-Baum betrifft. Rendezvousbasierte Verfahren führen dagegen zu Kontrollbäumen, die nicht am Routing-Baum für Quittungen und Übertragungswiederholungen ausgerichtet sind und deshalb zu erhöhtem Nachrichtenaufwand und zusätzlichen Verzögerungen führen, ganz zu Schweigen von der Verkehrskonzentration durch ERS-Suchnachrichten an der Rendezvous-Stelle. Werden dagegen senderbasierte Routing-Verfahren eingesetzt, so ergeben sich zwar kurze Kommunikationswege im Kontrollbaum, andererseits führt dies zu einem beträchtlichen Aufwand auf der Vermittlungsschicht. Jeder neue Multicast-Teilnehmer ist durch die ERS-Benutzung ein neuer Sender, für den ein eigener Spannbaum für das Routing aufgebaut wird. Aus der Beschreibung von DVMRP in Abschnitt 2.4.1.2 wurde ersichtlich, dass dieser Aufwand enorm ist. Aufgrund dieses Wissens kann bereits vermutet

werden, dass insgesamt der Nachrichtenaufwand für ERS und die Skalierbarkeit unbefriedigend sind. In Abschnitt 3.10 wird diese Vermutung bestätigt mit der Folgerung, dass ERS nicht für große Gruppen geeignet ist.

Der Zeitbedarf für den Gruppeneintritt mit ERS hängt davon ab, wie groß die Entfernung (Anzahl Teilstrecken) zwischen dem anmeldenden Knoten und dem nächsten Vaterknoten im Kontrollbaum ist sowie von der Wartezeit zwischen zwei Suchnachrichten. In Abschnitt 3.10 wird gezeigt, dass diese Wartezeit mit ERS nicht beliebig verringert werden kann, da sonst das Nachrichtenaufkommen explosionsartig zunimmt.

Die Orientierung des Kontrollbaums am Routing-Baum wurde bereits oben kurz diskutiert. Kurze Kommunikationswege im Kontrollbaum können nur mit senderbasierten Routing-Verfahren erreicht werden. Die für eine effiziente lokale Fehlerbehebung notwendige Eigenschaft, dass ein Gruppenführer näher am Sender liegt als seine Kindknoten, wird nicht erreicht. Für die Zuverlässigkeit des entstehenden Kontrollbaums führt ERS keinerlei Optimierungen durch.

Das ERA-Verfahren teilt einige Nachteile des ERS-Verfahrens. So ist auch hierzu ein Multicast-Dienst auf der Vermittlungsschicht notwendig und senderspezifisches PIM nicht ausreichend. Bezüglich der universellen Einsetzbarkeit ergeben sich dennoch Vorteile für ERA, da kein bidirektionales Netzwerk für dessen Einsatz notwendig ist. Probleme mit rendezvousbasierten Routing-Verfahren treten bei ERA in gleicher Weise auf wie sie bereits für ERS ausgeführt wurden. Auch das Problem eines separaten Routing-Baums für jeden neuen Teilnehmer bei senderbasierten Verfahren wird durch ERA nicht entschärft. Zwar ist dieser für die Anmeldung nicht erforderlich, aber sobald die Anmeldung erfolgt ist und der neue Teilnehmer selbst Kindknoten akzeptieren kann, muss ein eigener Routing-Baum aufgebaut werden. Deshalb und aus Gründen, die in Abschnitt 3.10 durch die Simulationsergebnisse gezeigt werden, ist auch der Nachrichtenaufwand und die Skalierbarkeit für ERA nicht akzeptabel. Vor allem besteht bei ERA das Problem, dass Suchnachrichten nach neuen Kindknoten periodisch und nicht bei Bedarf erfolgen, was zu einem unnötig hohen Nachrichtenaufwand führt. Von der Wartezeit zwischen dem Aussenden von Suchnachrichten hängt schließlich ebenfalls der Zeitbedarf für den Gruppenbeitritt ab. Es muss dabei abgewogen werden zwischen einem höheren Nachrichtenaufwand, aber einem schnelleren Kontrollbaumbeitritt und einem geringeren Nachrichtenaufwand, aber einem langsameren Kontrollbaumbeitritt.

Für die Orientierung des Kontrollbaums am Routing-Baum verhält sich ERA identisch zu ERS. Nur mit senderbasiertem Routing ergeben sich kurze Kommunikationswege im Kontrollbaum, während dies mit rendezvousbasiertem Routing nicht erreicht werden kann. Eine effiziente lokale Fehlerbehebung dadurch, dass Gruppenführer näher am Sender liegen als ihre Kindknoten, wird nicht erreicht. Bezüglich der Optimierung des Kontrollbaums nach seiner Zuverlässigkeit werden im ERA-Verfahren ebenfalls keine Maßnahmen getroffen.

Die Beurteilung der Router-basierten Verfahren ist diffiziler, da die Mechanismen eng mit der Fehlerkontrolle und –behebung verwoben sind. Der offensichtlichste und schwerwiegenste Nachteil von Router-basierten Verfahren ist deren Notwendigkeit, modifizierte Router einzusetzen (siehe [Cain & Towsley 2000, Cain et al. 2000] für Router-Erweiterungen). Dadurch ist nicht nur deren universelle Einsetzbarkeit stark eingeschränkt, viel mehr können diese Verfahren gegenwärtig im Internet nicht eingesetzt werden. Zwar kommen einige dieser Verfahren ebenfalls damit zurecht, wenn lediglich eine Teilmenge der Router entsprechend ihren Vorgaben modifiziert wurde, aber deren Leistung und Skalierbarkeit ist dann stark eingeschränkt. Und die Erfahrungen mit früheren Vorschlägen zur Modifizierung von Routern (z.B. IPv6) haben gezeigt, dass selbst bei überzeugenden Argumenten dafür, eine derartige Umstellung nur besonders langsam durchzusetzen ist. Weitere spezielle Kritikpunkte ergeben sich aus impliziten Annahmen mancher dieser Verfahren, z.B. ARM, dass mittels Unicast gesendete Quittungen den selben Weg zurück nehmen, den die Multicast-Datenpakete genommen haben. Diese Annahme ist nicht in jedem Fall berechtigt. Bei rendezvousbasierten Verfahren stellt diese Annahme sogar die Ausnahme dar, sodass theoretische Vorteile dieser Verfahren in der Praxis nicht immer erreicht werden. Obwohl manche dieser Verfahren keinen explizit aufgebauten Kontrollbaum benötigen, bedeutet dies nicht, dass dafür keine zusätzliche Netzlast erzeugt wird. Dies haben die obigen Ausführungen zu LMS und OTERS gezeigt. Generell kann zudem die zusätzliche Last bei den Routern kritisiert werden, die deren Skalierbarkeit einschränken. Gegenwärtig konzentrieren sich alle Anstrengungen darauf, die Router zu beschleunigen, um die Bandbreite kommender Netzgenerationen nicht zu begrenzen und gleichzeitig preiswertere Geräte mit weniger Spezial-Hardware herzustellen. Beide Anstrengungen sprechen dagegen, Router-basierte Dienste zu etablieren.

In den folgenden Abschnitten wird ein neues Verfahren zum Aufbau von Kontrollbäumen vorgestellt, das Token-Repository-Service (TRS) genannt wird. Dieses besitzt eine Reihe von Vorteilen gegenüber den bisherigen Verfahren:

- TRS kann vollkommen unabhängig von der Vermittlungsschicht arbeiten. Dies resultiert in einer universellen Einsetzbarkeit, da kein Multicast-Dienst und somit auch keine speziellen Eigenschaften einer Multicast-Unterstützung vorausgesetzt werden. Damit ist eine notwendige Voraussetzung gegeben, um den Dienst flächendeckend und rasch zu etablieren.

- TRS erzeugt lediglich eine geringe Netzlast und bietet Skalierbarkeit für sehr große Gruppen– und Teilnehmerzahlen.

- Der Zeitbedarf für den Gruppeneintritt ist gering und unabhängig von Wartezeiteinstellungen oder Abwägungen zwischen Netzlast und Zeitbedarf.

- Der aufgebaute Kontrollbaum orientiert sich an dem Router-Spannbaum, was für geringe Netzbelastung und niedrige Verzögerungszeiten für Übertragungswiederholungen sorgt.

- TRS ist sehr flexibel konfigurierbar und zudem an unterschiedlichste Anforderungen anpassbar. So kann die Zuverlässigkeit und der Reorganisationsaufwand des Kontrollbaums optimiert werden, eine Optimierung für Gruppen mit lediglich einem Sender durchgeführt werden oder die Verzögerung minimiert werden. Kombinationen davon sind ebenfalls möglich.

- TRS ist unabhängig vom zuverlässigen Transportprotokoll und lässt sich leicht in beliebige Transportprotokolle integrieren.

## 3.3 Generische Struktur hierarchischer Multicast-Transportprotokolle

Der Beschreibung von TRS sollen die Annahmen über die Struktur hierarchischer Transportprotokolle vorausgeschickt werden. Die hier gewählte Struktur ist sehr allgemein und erlaubt die Zusammenarbeit mit allen zuverlässigen Multicast-Protokollen, die ohne Router-Unterstützung vollständig auf Transportschicht arbeiten.

Vorausgesetzt werden autonome und asynchrone Rechnerknoten, die durch ein Kommunikationsnetzwerk miteinander verbunden sind. Zwischen beliebigen Paaren von Rechnern können Nachrichten ausgetauscht werden. Eine nichtleere Teilmenge der Rechnerknoten bildet eine Multicast-Gruppe. Es können beliebig viele Gruppen existieren. Multicast-Gruppen müssen nicht disjunkt sein. Ein Recherknoten kann folglich Mitglied in beliebig vielen Gruppen sein.

Der Kontrollbaum hierarchischer Multicast-Protokolle ist aus den Gruppenmitgliedern aufgebaut. Ein beliebiger Rechnerknoten kann ein Sender für eine Multicast-Gruppe sein. Die Anzahl der Sender pro Multicast-Gruppe kann größer als eins sein, wie es beispielsweise in Lorax [Levine & Garcia-Luna-Aceves 1998] oder RMTP-II [Whetten & Taskale 2000] der Fall ist. Der Sender muss nicht selbst zur Multicast-Gruppe gehören. Beispielsweise gehört der Sender in TMTP der Gruppe an, wohingegen dies in RMTP nicht zwingend notwendig ist, da Quittungen mittels Unicast an den Sender übertragen werden. In Abbildung 3.2 ist ein Beispiel für die möglichen Szenarien gegeben. Ein Sender kann auch ein Empfänger sein (z.B. Lorax), kann die Wurzel des Kontrollbaums sein (z.B. RMTP und TMTP, siehe Abschnitt 2.5.3) oder kann außerhalb des Kontrollbaums stehen (z.B. RMTP-II). Für einen einzigen Sender empfiehlt es sich, dass der Sender auch gleichzeitig die Wurzel des Kontrollbaums ist, wird aber nicht vorausgesetzt.

Eine auszuliefernde Nachricht kann vom Sender direkt mittels eines Multicast-Diensts an die Gruppe gesendet werden, wie es beispielsweise in RMTP-II vorgesehen ist. Das Senden per Unicast an die Kontrollbaumwurzel soll allerdings nicht ausgeschlossen werden. Die Kontrollbaumwurzel kann die Nachricht mittels eines Multicast-Diensts ausliefern oder aber mittels Unicast über den Kontrollbaum

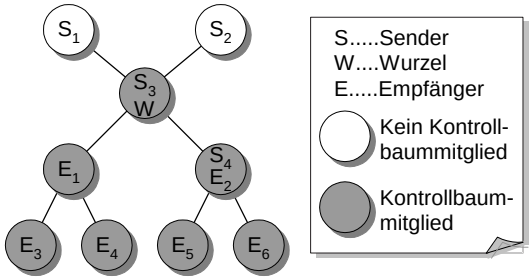



*Abbildung 3.2: Mögliche Sender einer Multicast-Gruppe*

verteilen. In diesem Fall ist kein Multicast-Dienst auf Vermittlungsschicht notwendig. In der nachfolgenden Beschreibung wird allerdings zur Vereinfachung der Beschreibung davon ausgegangen, dass IP-Multicast verfügbar ist, wie es die gängige Annahme für zuverlässige Transportprotokolle darstellt.

Die Quittungen sollen im Kontrollbaum immer in Richtung der Wurzel gesammelt werden. Ist die Wurzel nicht gleichzeitig der Sender der Nachricht (z.B. in RMTP-II), teilt sie schließlich als letzter Schritt dem eigentlichen Sender der Nachricht die erfolgreiche Auslieferung mit oder verlangt nach einer Übertragungswiederholung, wenn die Wurzel die Nachricht selbst nicht erfolgreich empfangen hat.

Diese Beschreibung definiert ein sehr allgemeines Modell der Multicast-Auslieferung. Die Notwendigkeit ergibt sich daraus, dass alle Multicast-Transportprotokolle durch das Modell abgedeckt werden sollen.

## 3.4 Konzept und Schnittstelle des Token-Repository-Service

Anstatt das Netzwerk nach einem passenden Gruppenführer im Kontrollbaum abzusuchen, stellt der TRS-Ansatz einen Verzeichnisdienst dar, ähnlich dem Domain-Name-Service [Mockapetris 1987a, Mockapetris 1987b]. Dieser Dienst wird den Multicast-Protokollen zur Verfügung gestellt. Ein neuer Teilnehmer fragt beim TRS-Dienst nach einem passenden Gruppenführer. Dieser wird dem neuen Teilnehmer genannt, der sich beim referenzierten Gruppenführer als Kindknoten anmelden kann. Die eigentliche Anmeldung wird dabei nicht durch den TRS-Dienst, sondern durch das Multicast-Transportprotokoll vorgenommen. In Abbildung 3.3 ist die Schnittstelle zwischen den Multicast-Teilnehmern und dem TRS-Dienst aufgezeigt. Die Schnittstelle zwischen dem Multicast-Transportprotokoll und den Teilnehmern am Multicast-Dienst ist nicht Gegenstand dieser Abhandlung. Dafür sind allein die Multicast-Transportprotokolle zuständig (siehe z.B. [Whetten et al. 2001]). Daher wird im Folgenden die Schnittstelle zwischen dem TRS-Dienst und den Teilnehmern beschrieben.

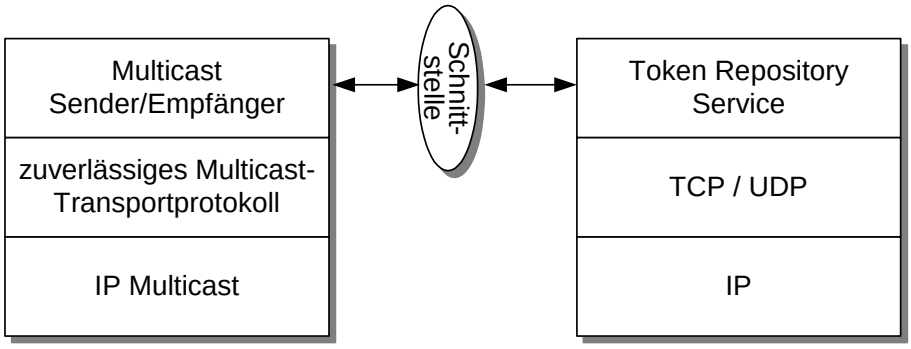



*Abbildung 3.3: Schichtenmodell und Schnittstelle zwischen den Multicast-Teilnehmern und TRS*

Für eine genauere Beschreibung des Diensts und der Schnittstelle wird das Konzept der *Marke* (engl. Token, daher Token-Repository-Service) eingeführt. Eine Marke repräsentiert für einen neuen Teilnehmer das Recht sich an einen bestimmten Gruppenführer im Kontrollbaum anzumelden. Nachdem ein Wurzelknoten mit maximalem Verzweigungsgrad *B* eine Gruppe erstellt, werden ebenfalls *B* Marken erstellt und im TRS abgelegt. Der erzeugende Knoten wird *Eigentümer* der Marken genannt. Eine Marke ist ein Tripel <*Gruppe*, *Eigentümer*, *Metrikbetrag*>. *Gruppe* bezeichnet die Multicast-Adresse der Gruppe, *Eigentümer* ist der Gruppenführer im Kontrollbaum, an den sich ein neuer Teilnehmer anmelden kann und in *Metrikbetrag* werden weitere Eigenschaften der Marke abgelegt.

Mit Hilfe der Metrik ist es möglich, eine bezüglich der Metrik optimale Marke zu ermitteln und nicht nur eine, die lediglich der gesuchten Multicast-Gruppe angehört. Beispiele für Metriken sind der Abstand des Eigentümers zum Wurzelknoten, sowohl als Distanz im Netzwerk als auch als Distanz im logischen Kontrollbaum betrachtet, oder die Verzögerung im Netzwerk für eine Nachricht zwischen Wurzelknoten und Eigentümer. Die Bedeutung der Metriken wird in Abschnitt 3.5.2 näher erläutert.

Initial sind *B* Marken im TRS vorhanden, die der Wurzelknoten einer Multicast-Gruppe erzeugte. Wenn nun ein weiterer Teilnehmer, der *T* genannt werden soll, in den Kontrollbaum eintreten möchte, muss der TRS-Dienst nach einer passenden Marke befragt werden. Innerhalb des TRS-Diensts wird eine Marke dieser Gruppe ausgewählt und an *T* gesendet. Mit Hilfe der Eigentümerinformation in der Marke kann sich *T* an einen Gruppenführer im Kontrollbaum anmelden.

Nun muss dafür Sorge getragen werden, dass die Marken im TRS-Dienst nicht nur verbraucht, sondern dass auch neue erstellt werden. Jeder neue Teilnehmer, der sich aus dem TRS-Dienst mit einer Marke versorgt, muss selbst wiederum als potentieller Vaterknoten, d.h. als Gruppenführer einer lokalen Gruppe im Kontrollbaum verfügbar sein. Dazu übergibt *T* bereits bei der Markenanfrage seinen maximalen Verzweigungsgrad, der nicht kleiner als eins sein darf. Anhand dessen wird die entsprechende Anzahl an Marken mit Eigentümer *T* im TRS-Dienst neu erstellt. Ohne diese Forderung wäre

es möglich, dass alle Marken im TRS-Dienst verbraucht werden und somit keine weiteren Anmeldungen durchführbar wären.[1]

Wenn ein Teilnehmer *T* eine Gruppe *G* verlassen möchte, dann werden alle Marken im TRS-Dienst mit der Gruppe *G* und dem Eigentümer *T* gelöscht, damit keine Anmeldungen mehr möglich sind. Die Annahme dabei ist, dass ein Teilnehmer, der den Kontrollbaum verlässt, keine Kindknoten mehr besitzt. Gegebenenfalls muss das Multicast-Transportprotokoll dafür Sorge tragen, dass sich die Kindknoten an anderer Stelle erneut anmelden.[2] Neben den Marken für mögliche Kindknoten hat zudem der verlassende Teilnehmer selbst eine Marke reserviert, die für die Anmeldung an seinen Gruppenführer notwendig war. Diese Marke wird an den TRS-Dienst zurückgegeben, so dass sie von einem anderen Teilnehmer verwendet werden kann.

Zusammengefasst stellt der TRS-Dienst die folgenden Operationen den Teilnehmern zur Verfügung:

1. repCreateGroup(G, N, B)
   Anlegen einer neuen Gruppe *G*. Der Knoten *N* wird zum Wurzelknoten des Kontrollbaums, der einen maximalen Verzweigungsgrad von *B* besitzt.

2. repDeleteGroup(G)
   Auflösen einer Gruppe *G*. Löschen aller Marken, die zur Gruppe *G* gehören.

3. repJoinGroup(G, N, B): T
   Ein neues Mitglied *N* fragt nach einer Marke *T* der Gruppe *G*. Nach der Übergabe der Marke ist *N* selbst in der Lage für Kindknoten als Gruppenführer zu agieren. Der maximale Verzweigungsgrad von *N* im Kontrollbaum ist *B*, mit $B > 0$.

4. repLeaveGroup(G, M)
   Gruppenmitglied *M* verlässt die Gruppe *G*. Löschen aller zur Gruppe *G* gehörenden Marken, deren Eigentümer *M* ist.

5. repAddToken(G, E)
   Hinzufügen einer neuen Marke mit Eigentümer *E*. Diese Operation wird aufgerufen, wenn ein Kindknoten von *E* den Kontrollbaum verlässt.

[1] Offensichtlich kann es auch Multicast-Teilnehmer geben, die sich nicht für die Funktion eines Gruppenführers eignen. Dazu gehören beispielsweise Teilnehmer, die sehr unzuverlässig sind oder Teilnehmer, die nur kurze Zeit der Multicast-Gruppe angehören. Anstatt die Gruppenführerschaft durch einen maximalen Verzweigungsgrad von null auszuschließen, damit aber möglicherweise alle Marken einer Gruppe im TRS-Dienst zu verbrauchen, besteht die elegantere Möglichkeit darin, dies über den Metrikbetrag zu steuern (siehe auch Abschnitt 3.7).

[2] Diese Problematik wird beispielsweise in [Yavatkar et al. 1995] unkompliziert gelöst, indem Gruppenführer den Kontrollbaum nicht verlassen dürfen. Da Gruppenführer aber auch ausfallen können, sollte von dieser Lösung Abstand genommen werden. In [Levine et al. 1996] wird dagegen ein Algorithmus zum Reorganisieren des Kontrollbaums vorgestellt. Eine einfache Lösung besteht darin, die Kindknoten über den Gruppenaustritt zu informieren, die sich daraufhin mit Hilfe des TRS-Diensts neu anmelden.

6. repRefreshToken(G, E, A)
   Auffrischen der Markeninformation. Der Eigentümer *E* meldet die Anzahl der noch freien Kindknoten *A* im Kontrollbaum an den TRS-Dienst. Diese Funktionalität wird im Fehlerfall benötigt und später in Abschnitt 3.6 näher erläutert.

Mit der nunmehr festgelegten Schnittstelle können die Multicast-Teilnehmer auf den TRS-Dienst zugreifen. In den folgenden Abschnitten soll die bisher abstrakt gehaltene Beschreibung des TRS-Diensts detailliert werden.

## 3.5 Realisierung des Token-Repository-Service

### 3.5.1 Überblick

Der bisher aus der Sichtweise der Klienten als zentrales System beschriebene TRS-Dienst (lokationstranparent für Klienten) wird als ein verteiltes System bestehend aus *Token-Repository-Server*, kurz *RepServer*, realisiert. Diese Entwurfsentscheidung ist aus drei wichtigen Anforderungen entstanden. Erstens soll der Dienst skalierbar sein, d.h. auch für eine große Anzahl von Multicast-Gruppen und Teilnehmer geeignet sein. Zweitens soll der Dienst zuverlässig sein und keinen sogenannten Single-Point-of-Failure besitzen. Drittens sollen neue Teilnehmer Marken von Gruppenführer erhalten, die im Netzwerk nahe beieinander liegen, damit die Kommunikationsdistanzen im Kontrollbaum minimiert werden (siehe Abschnitt 3.2.3). Durch die verteilten RepServer kann diese Forderung ermöglicht werden, wie später erläutert wird.

Jeder RepServer ist für eine *Domäne* verantwortlich. Domänen strukturieren das Netzwerk in disjunkte Mengen von Knoten. Dabei sollen die Kommunikationsdistanzen von Knoten innerhalb einer Domäne möglichst klein sein, d.h. die Kommunikationsdistanz zwischen zwei Knoten in derselben Domäne ist typischerweise kleiner wie zwischen zwei Knoten in verschiedenen Domänen. Als Kommunikationsdistanz soll primär die Anzahl der Router-Teilstrecken (engl. hops) betrachtet werden, die möglichst klein sein sollte. Aber auch eine Strukturierung nach der Verzögerung kann sinnvoll sein. So kann gegebenenfalls ein Satellitennetzwerk mit höherer Verzögerung eine eigene Domäne bilden. Die Forderung nach geringen Kommunikationsdistanzen innerhalb von Domänen führt zu geringen Kommunikationsdistanzen in lokalen Gruppen, d.h. zwischen Vater– und Kindknoten innerhalb des Kontrollbaums.

In Abbildung 3.4 ist eine Strukturierung des Netzwerks nach der Kommunikationsdistanz, bezüglich der Anzahl Teilstrecken, durchgeführt worden. RepServer $S_1$ ist verantwortlich für Domäne 1, bestehend aus allen Knoten $N_{1X}$. In Abbildung 3.4 sind alle RepServer selbst Mitglied ihrer verwalteten Domäne. Dies ist keine notwendige Voraussetzung. Prinzipiell können die RepServer an

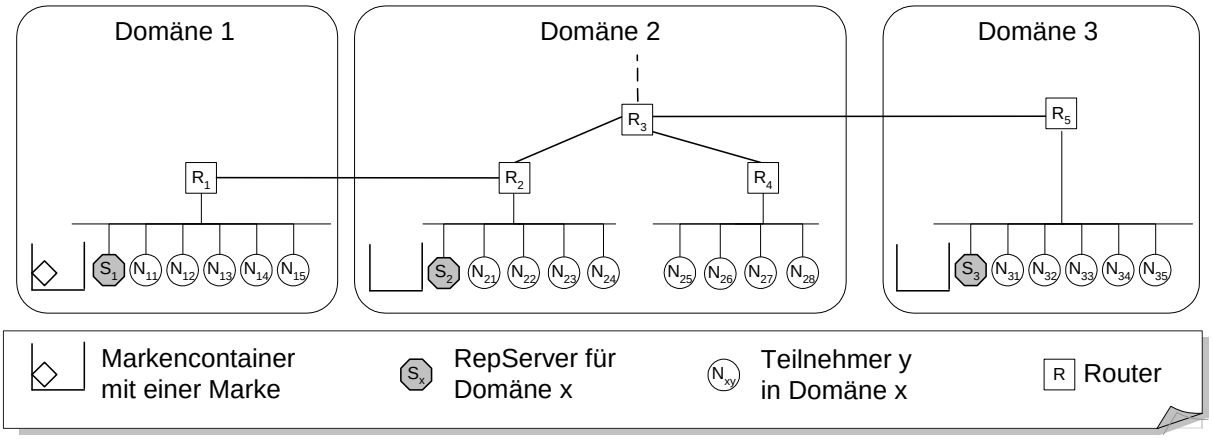



*Abbildung 3.4: Domäneneinteilung eines Netzwerks*

beliebiger Stelle platziert sein. Der Vorteil einer Platzierung innerhalb ihrer eigenen Domäne liegt in den niedrigen Kommunikationskosten und –verzögerungen beim Zugriff der Multicast-Teilnehmer auf den RepServer. Aus Sicht der Zuverlässigkeit ergibt sich ein weiterer Vorteil. Eine kleine Distanz zwischen Knoten und RepServer minimiert die Wahrscheinlichkeit von Nachrichtenverlust und einer Netzpartitionierung.

Jeder Knoten im Netzwerk kennt seinen *lokalen RepServer*, d.h. den RepServer seiner Domäne, beispielsweise durch geeignete Konfiguration ähnlich dem Domain-Name-Service [Mockapetris 1987a] oder ermittelt den lokalen RepServer mit einem Anycast-Dienst [Partridge et al. 1993]. In Abbildung 3.4 ist $S_1$ der lokale RepServer für alle Knoten $N_{1X}$. Mit der Ausnahme von Fehlersituationen erfolgt der Zugriff auf den TRS-Dienst immer über den lokalen RepServer.

Alle Marken werden auf den verteilten RepServern gespeichert und zwar jeweils bei dem RepServer, der die Marken erzeugt hat. Dies vereinfacht das Löschen von Marken bei Abmeldungen, da der Speicherort bekannt ist. Für Abbildung 3.4 bedeutet dies, dass $S_1$ alle Marken speichert, die durch das Erstellen von Gruppen oder das Anmelden seiner lokalen Knoten $N_{1X}$ erzeugt wurden. Zu beachten gilt, dass durch diese Speicherart ein RepServer üblicherweise nicht Marken für alle existierenden Gruppen speichert.

Wenn ein neuer Teilnehmer eine Anfrage nach einer Marke an seinen lokalen RepServer stellt, so ist der einfachste Fall, dass der lokale RepServer über eine Marke dieser Gruppe verfügt. Der RepServer übergibt eine Marke an den neuen Teilnehmer und löscht diese aus seiner Datenbasis. Wurde bei einem RepServer hingegen die Gruppe weder angelegt noch eine vorherige Anmeldung durchgeführt, so besitzt dieser keine Marken. Sofern keine Fehler auftreten, dieses wird später in Abschnitt 3.6 behandelt, ist allerdings immer mindestens ein RepServer vorhanden, der Marken der gesuchten Gruppe besitzt. Beispielsweise ist in Abbildung 3.4 die Situation dargestellt, in der lediglich RepServer $S_1$ eine Marke besitzt. Wenn nun ein Knoten aus Domäne 2 seinen lokalen RepServer $S_2$ nach einer

Marke fragt, kann diese Anfrage nicht direkt beantwortet werden. Um dennoch eine Marke liefern zu können, wird eine Markensuche über die Menge aller RepServer initiiert, um schließlich eine Marke von RepServer $S_1$ übergeben zu können. Für die effiziente Suche einer Marke wurden verschiedene Strategien entwickelt, die in den Abschnitten 3.5.4 – 3.5.6 detailliert vorgestellt werden.

Eine erste Strategie für die Markensuche hat zum Ziel, die Implementierung des TRS-Diensts möglichst einfach zu realisieren. Eine Markensuche wird durch eine Multicast-basierte erweiternde Ringsuche durchgeführt. Dieser Ansatz wird daher TRS-R genannt (siehe [Maihöfer 2000b, Maihöfer 2000c]). Alle RepServer gehören der selben, wohlbekannten Multicast-Gruppe an. Eine Markensuche wird im Gegensatz zu herkömmlichem ERS (siehe Abschnitt 3.2.1) nicht auf der Multicast-Adresse der Gruppe durchgeführt, an welche die Anmeldung erfolgen soll, sondern auf der Multicast-Adresse aller RepServer. Wenn die erweiternde Ringsuche einen RepServer erreicht, der eine Marke der gesuchten Gruppe besitzt, dann kann diese an den suchenden RepServer übergeben werden. Durch die geringe Anzahl an RepServern im Vergleich zur Anzahl der Teilnehmer in großen Multicast-Gruppen und dadurch, dass jeder RepServer pro Gruppe maximal einmal eine Markensuche durchführen muss, ist dieser Ansatz bereits wesentlich effizienter als herkömmliches ERS.

Neben der möglichst einfachen Implementierung ist eine Realisierung mit möglichst geringen Kosten für die Markensuche sinnvoll. Dieser Ansatz wird TRS-K genannt (siehe [Maihöfer & Rothermel 1999a, Rothermel & Maihöfer 1999]). Wichtig ist dabei die Feststellung, dass die Schnittstelle zwischen den Multicast-Teilnehmern und dem TRS-Dienst beibehalten wird, d.h. es ist ohne Änderung der Schnittstellen möglich, transparent für die Klienten vom einfacheren TRS-R zum effizienteren TRS-K zu wechseln.

Um die Suchkosten zu minimieren wird bei TRS-K keine Ringsuche mehr eingesetzt. Stattdessen werden alle RepServer in einer Hierarchie strukturiert. Die Markensuche kann als Folge davon sehr effizient innerhalb des Baums durchgeführt werden. Jeder innere RepServer des Baums und die Wurzel speichern Informationen, ob die Kind-RepServer Marken der verschiedenen Gruppen besitzen. Nur die Blatt-RepServer speichern Marken. Deshalb werden alle anderen RepServer als Such-RepServer bezeichnet. Mit dieser Organisation kann in einem Durchlauf des Baums, ausgehend von der Wurzel bis zu einem Blatt-RepServer, in wenigen Schritten ein Halter einer Marke bestimmt werden. Zu beachten ist, dass diese Organisations– und Suchform analog zu allgemeinen Such– und Sortieralgorithmen ist, bei denen sich eine hierarchische Organisation, mit dem daraus resultierenden logarithmischen Suchaufwand, bei vielen Algorithmen bewährt hat [Sedgewick 1991]. Auch in verteilten Systemen wird dies vorteilhaft genutzt, z.B. bei DNS [Mockapetris 1987a, Mockapetris 1987b] oder der weltweiten Lokalisierung verteilter Objekte [van Steen et al. 1996].

Um einen Flaschenhals an der Wurzel zu verhindern, startet eine Markensuche nicht beim Wurzel-RepServer, sondern immer bei einem Blatt-RepServer. Ganz allgemein erfolgt der Zugriff der Klienten auf den TRS-Dienst stets über die Blatt-RepServer. Ist dort keine Marke vorhanden, so wird

eine Markensuche in Richtung Wurzel gestartet. Die meisten Markensuchen stoßen bereits vor dem Erreichen des Wurzel-RepServers auf einen RepServer, der die Richtung der Markensuche umkehrt, da in einem Teilbaum Marken gespeichert sind.

Schließlich wurde aufbauend auf TRS-K eine weitere Strategie entwickelt, die bei der Markensuche Metriken berücksichtigen kann und somit zu optimalen Kontrollbäumen innerhalb der gewählten Metrik führt. Dieser Ansatz wird TRS-M genannt (siehe [Maihöfer & Rothermel 1999b, Maihöfer 2001]). Auch bei TRS-M sind die RepServer in einem Suchbaum organisiert. Anstatt bei den Such-RepServern lediglich einen Bitvektor zu speichern, der für jeden Unterbaum die Angabe enthält, ob eine Marke vorhanden ist, wird ein komplexerer Vektor benötigt. Dieser enthält zusätzlich für jeden Unterbaum den Metrikbetrag der optimalen Marke innerhalb dieser Metrik. Bei einer Markensuche kann nun aufgrund der gespeicherten Metrikbeträge nicht nur eine beliebige Marke gefunden werden, sondern bezüglich dieser Metrik sogar eine optimale. Ein Beispiel für eine Metrik ist die Höhe des Markeneigentümers im Kontrollbaum, d.h. der Abstand zur Wurzel (siehe Abschnitt 3.5.2). Wird nach dieser Metrik optimiert, können höhenbalancierte Kontrollbäume erstellt werden, welche die höchste Zuverlässigkeit bieten. Die Definition eines höhenbalancierten Kontrollbaums und Erläuterungen zur Zuverlässigkeit eines Kontrollbaums erfolgen im nächsten Abschnitt. Ein weiteres Beispiel für eine Metrik ist die Verzögerung des Markeneigentümers zum Wurzelknoten. Mit Hilfe dieser Metrik können die Verzögerungszeiten für Übertragungswiederholungen minimiert werden.

Alle Suchstrategien garantieren, dass immer ein Markeneigentümer ausgewählt wird, der möglichst nahe (bezüglich der Anzahl der Teilstrecken oder der Verzögerung, je nach gewählter Domänenstrukturierung) zum neuen Teilnehmer platziert ist. Für TRS-M ergibt sich die vorrangige Optimierung bezüglich der Metrik eine optimale Marke zu finden. Existieren gleichwertige Marken, wird der nächstgelegene Markeneigentümer gewählt. Stehen innerhalb einer Domäne mehrere Marken zur Auswahl, dann wird bei TRS-R und TRS-K ebenfalls die Marke mit dem optimalen Metrikbetrag gewählt, allerdings nur bezogen auf diese Domäne. Zu beachten gilt, dass der Metrikbetrag einer Marke bei allen drei Suchstrategien bekannt ist. Allerdings erlaubt nur TRS-M die domänenübergreifende Suche bezüglich der Metrik.

### 3.5.2 Struktur der Marken

Ein RepServer speichert alle Marken, die zur selben Gruppe gehören, in einem *Markencontainer*. Ein Markencontainer besitzt folgende Struktur:

1. Gruppenbezeichner *GR*: Eindeutiger Bezeichner der Multicast-Gruppe, z.B. durch die IP-Multicast-Adresse.

2. Paketmenge *SP*: Menge von Markenpaketen der selben Multicast-Gruppe.

Innerhalb eines Markencontainers sind *Markenpakete* gespeichert. Ein Markenpaket beinhaltet alle Marken, die nicht nur der selben Gruppe angehören, sondern auch den selben Eigentümer besitzen. Die Struktur eines Markenpakets ist:

1. Eigentümer $E$: Eindeutiger Bezeichner des Markeneigentümers, der als Gruppenführer im Kontrollbaum für einen neuen Teilnehmer fungiert.
2. Metrikbetrag $M$: Der Metrikbetrag erlaubt die Suche nach optimalen Marken bezüglich dieser Metrik.
3. Anzahl $A$: Die Anzahl verfügbarer Marken in diesem Markenpaket.
4. Ablaufzeit $AZ$: Zeitpunkt, zu dem das Markenpaket ungültig wird.

Ein Markenpaket wird angelegt, wenn die ersten Marken auf einem RepServer gespeichert werden. Nachdem die letzte Marke in einem Markenpaket gelöscht wurde, wird auch das Markenpaket gelöscht. Dasselbe gilt für einen Markencontainer.

Die Bedeutung der Metrik bedarf noch ausführlicherer Erläuterung. Prinzipiell kann der Metrikbetrag beliebig geartet sein, also ein Skalar oder auch ein Vektor sein, der seinerseits mehrere verschiedene Metrikbeträge vereint. Es muss sich allerdings mindestens um eine Ordinalskala handeln, damit eine Vergleichsfunktion Metrikbeträge bewerten kann, um die optimale zu ermitteln. Eine Halbordnung ist dafür ausreichend, d.h. gleiche Beträge sind erlaubt. Für die einfachere Beschreibung der Algorithmen sollen zwei einschränkende Annahmen getroffen werden, die keine Beschränkung der Allgemeinheit darstellen. Im Folgenden handelt es sich bei dem Metrikbetrag um einen skalaren Wert aus dem positiven natürlichen Zahlenbereich und zweitens ist der beste Metrikbetrag durch den Wert eins dargestellt. Ein höherer Metrikbetrag entspricht einem schlechteren Wert.

Folgendes Beispiel, das durchgängig für die Beschreibung der Algorithmen Verwendung findet, soll den möglichen Einsatz der Metrik verdeutlichen. Eine Optimierung der Markenauswahl kann nach der Höhe des Markeneigentümers im Kontrollbaum erfolgen. Das heißt, es werden vorwiegend Marken gewählt, deren Eigentümer eine möglichst kurze Distanz im Kontrollbaum zum Wurzelknoten besitzen. Diese Metrik soll darum als Höhe des Markeneigentümers im Kontrollbaum, kurz Markenhöhe bezeichnet werden. Unter der Annahme, dass die Wurzel die Höhe eins besitzt, sind die Höhen in Abbildung 3.5 beispielhaft dargestellt.

Wird keine Optimierung der Markenauswahl nach der Markenhöhe vorgenommen, dann entstehen Kontrollbäume mit sehr vielen freien Plätzen für mögliche Kindknoten an Gruppenführern. Die linke Seite der Abbildung 3.5 zeigt hierfür ein Beispiel. Die Baumhöhe des Kontrollbaums sei definiert als die maximale Höhe aller Knoten im Kontrollbaum, d.h. $max(\forall\, n \in \text{Knoten: Höhe von Knoten } n)$. Aus

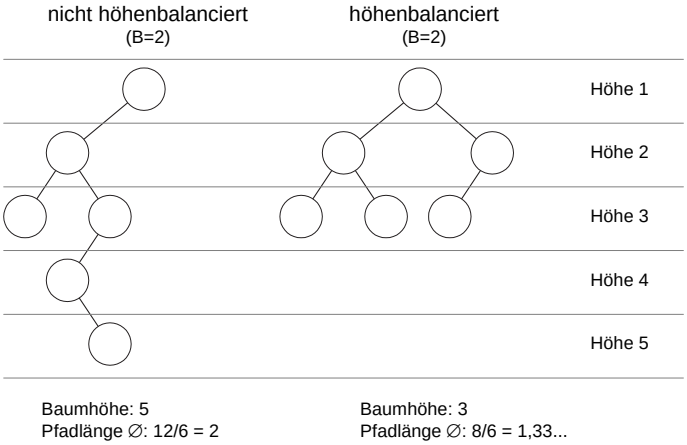



*Abbildung 3.5: Höhenbalancierte Kontrollbäume*

der Abbildung wird gleichfalls ersichtlich, dass dies eine größere Baumhöhe nach sich zieht. Verbunden damit sind mehrere Nachteile: Eine nicht minimale Baumhöhe (1) verursacht einen größeren Reorganisationsaufwand, wenn Knoten den Kontrollbaum verlassen, (2) führt zu größeren Pfadlängen im Kontrollbaum und folglich zu geringerer Zuverlässigkeit und schließlich (3) zu größerer Verzögerung beim Senden von Übertragungswiederholungen und zu größerer Verzögerung beim Sammeln der aggregierten Quittungen. Die Nachteile (1) und (2) ergeben sich aus dem ungünstigen Verhältnis zwischen Gruppenführern und allen anderen Knoten, genannt *Blattknoten*, da nicht minimale Baumhöhen zu mehr Gruppenführern und weniger Blattknoten führen. Verlässt ein Gruppenführer den Kontrollbaum, ist damit ein beträchtlicher Reorganisationsaufwand verbunden, da sich alle Knoten dieses Teilbaums mit dem verlassenden Teilnehmer als Wurzel erneut beim Kontrollbaum anmelden müssen. Erfährt ein Gruppenführer einen Ausfall, dann ist davon ebenso der gesamte Teilbaum mit dem Gruppenführer als Wurzel betroffen, was ebenfalls zu einem großen Reorganisationsaufwand und zur Minderleistung des Transportprotokolls durch häufige Übertragungswiederholungen während dieser Zeitspanne führt. Eine größere Gruppenführeranzahl erhöht die Wahrscheinlichkeit für eine derartige Situation und verringert damit die Zuverlässigkeit des Kontrollbaums. Schließlich werden durch die größeren Pfadlängen auch die Verzögerungen für Übertragungswiederholungen im Kontrollbaum erhöht, da diese im Extremfall über alle Hierarchiestufen erfolgen müssen (siehe dazu auch Abschnitt 4.5.4).

Die beschriebenen Nachteile werden durch höhenbalancierte Kontrollbäume minimiert. Ein höhenbalancierter Kontrollbaum besitzt die Baumhöhe $h = \lceil log_B(R(B-1)+1) \rceil$, wobei $B$ der maximale Verzweigungsgrad ist und $R$ die Teilnehmerzahl der Multicast-Gruppe. Die Herleitung dieser Formel wird in Abschnitt 3.9.2 erfolgen. Die rechte Seite der Abbildung 3.5 zeigt einen höhenbalancierten Kontrollbaum. Mit einem maximalen Verzweigungsgrad von $B = 2$ ergibt sich für die Kontrollbaumhöhe $h = \lceil log_2(6(2-1)+1) \rceil = \lceil 2.807\ldots \rceil = 3$.

Neben der Metrik bedarf zudem die Ablaufzeit der Marken einer genaueren Erläuterung. Die gesamte Markeninformation wird nach dem Soft-State-Prinzip verwaltet [Clark 1988]. Sowohl die Marken als auch später die Gruppenregister (siehe Abschnitt 3.5.5) sind mit einer Ablaufzeit verknüpft. Wird die Ablaufzeit vor dem Ende nicht verlängert, dann verfällt die zugehörige Information automatisch, d.h. die Marke wird gelöscht. Obwohl die in den nachfolgenden Abschnitten vorgestellten Protokolle nicht mehr aktuelle Informationen explizit löschen, sorgt der Soft-State-Ansatz für das automatische Löschen in Fehlersituationen, so dass die Robustheit des Protokolls erhöht wird. Dadurch kann oftmals auf aufwändigere Protokolle verzichtet werden. Doch wie kann die Ablaufzeit verlängert werden? Die Lebenszeit einer Marke ist korreliert mit der Lebenszeit des Eigentümers. Der Eigentümer sendet jeweils kurz vor dem Erreichen der Ablaufzeit und vorausgesetzt er gehört weiterhin dem Kontrollbaum an, eine Verlängerungsaufforderung an den zugehörigen RepServer. Die Ablaufzeit des Markenpakets wird daraufhin entsprechend verlängert. Ansonsten, wenn keine Verlängerungsaufforderung eintrifft, wird das Markenpaket gelöscht.

In den folgenden Beschreibungen der Algorithmen soll die Markenhöhe als Metrik verwendet werden. Die Strategien TRS-R und TRS-K können die Optimierung nach der Metrik nur lokal bei einem RepServer und folglich nur begrenzt durchführen. TRS-M ist dagegen in der Lage, eine globale Optimierung nach der Metrik über alle RepServer durchzuführen und dadurch höhenbalancierte Kontrollbäume zu erzeugen.

### 3.5.3 Systemmodell

Für die Realisierung des TRS-Diensts wird das folgende Systemmodell angenommen. Vorausgesetzt werden analog zur Beschreibung generischer Transportprotokolle in Abschnitt 3.3 autonome Rechnerknoten, die durch ein Kommunikationsnetzwerk miteinander verbunden sind. Zwischen beliebigen Paaren von Rechnern können Nachrichten ausgetauscht werden. Dies ist die einzige Möglichkeit der Kommunikation und Synchronization zwischen Rechnerknoten, da kein gemeinsamer Speicher vorhanden ist.

Eine nichtleere Teilmenge der Rechnerknoten kennzeichnet die RepServer und beliebige weitere nichtleere Teilmengen bilden die Multicast-Gruppen. Die Teilmengen müssen nicht disjunkt sein, d.h. ein Knoten kann RepServer sein und gleichzeitig mehreren Multicast-Gruppen angehören. Aus diesem Grund kann ein Knoten mehrere Prozesse nebenläufig ausführen.

Zur Vereinfachung der Protokollbeschreibung wird zunächst die Annahme getroffen, dass weder Knoten– noch Kommunikationsfehler auftreten. Eine Überlegung hinsichtlich stabilem oder flüchtigem Speicher erübrigt sich dadurch. In Abschnitt 3.6 werden diese vereinfachenden Annahmen aufgehoben und mit einer Beschreibung des Protokollverhaltens in Fehlersituationen ergänzt.

Die Uhren der Rechnerknoten müssen nicht synchronisiert sein. Einzig der Uhrendrift ist begrenzt und bekannt. Diese Forderung ist hinreichend, um die Ablaufzeiten, wie sie bereits im vorigen Abschnitt für die Marken eingeführt wurden, zu verwalten. Anstatt beim Setzen und Verlängern der Ablaufzeit den Zeitpunkt zu definieren wird dazu lediglich die Zeitspanne angegeben, was ohne synchronisierte Uhren erfolgen kann.

Für die TRS-R Strategie ist zudem erforderlich, dass ein unzuverlässiger Multicast-Dienst auf Vermittlungsschicht entsprechend dem Multicast-Dienstmodell, wie es in Abschnitt 2.2 beschrieben wurde, zur Verfügung steht.

### 3.5.4 Strategie 1: TRS mit erweiternder Ringsuche (TRS-R)

Im Folgenden wird die Strategie TRS-R im Detail erläutert. Dabei findet eine Unterscheidung hinsichtlich der vier Gruppenoperationen, Gruppenerstellung, Gruppenauflösung, Gruppenbeitritt und Gruppenaustritt, statt.

#### 3.5.4.1 Gruppenerstellung

Ein beliebiger Knoten initiiert die Gruppenerstellung mit einer *repCreateGroup(G,N,B)*-Operation an seinem lokalen RepServer *R*. In dessen Folge wird von *R* ein Markencontainer für die Gruppe *G* angelegt (siehe Pseudocode in Abbildung 3.6). In den Markencontainer wird ein Markenpaket mit Eigentümer *N* abgelegt. *B* spezifiziert die Anzahl der Marken im Markenpaket. Auch der Metrikbetrag und die Ablaufzeit werden dabei mit Werten belegt. Bezogen auf das Beispiel, die Höhe des Markeneigentümers im Kontrollbaum als Metrik zu verwenden, bedeutet dies, dass alle initial erstellten Marken den Metrikbetrag eins zugewiesen bekommen. Die Belegung der Ablaufzeit soll hier nicht näher ausgeführt werden. In Abschnitt 3.5.2 wurde das generelle Konzept bereits erläutert und in Abschnitt 3.6 wird darauf nochmals detaillierter Bezug genommen werden.

Am Beispiel von Abbildung 3.4 soll die Gruppenerstellung verdeutlicht werden. Knoten $N_{11}$ initiiert eine *repCreateGroup*-Operation an seinem lokalen RepServer $S_1$, der für Domäne 1 zuständig ist. Daraufhin legt $S_1$ einen Markencontainer an. In Abbildung 3.4 ist dieser bereits durch eine Marke mit Metrikbetrag eins dargestellt.

---

*R* **empfängt** repCreateGroup(*G*, *N*, *B*) **von** Klient *C*:
(1) erstelle Markenpaket mit *B* Marken für Eigentümer *N* und belege den initialen Metrikbetrag;
(2) erstelle Markencontainer für Gruppe *G* und füge das Markenpaket ein;

---

*Abbildung 3.6: Erstellen einer Gruppe in TRS-R*

#### 3.5.4.2 Gruppenauflösung

Die Operation *repDeleteGroup(G)* führt zum Löschen aller zur Gruppe *G* gehörenden Marken im gesamten TRS. Dazu löscht der aufgerufene RepServer seinen Markencontainer der Gruppe *G* und sendet eine *DeleteGroup(G)*-Nachricht an alle anderen RepServer. Da alle RepServer einer wohlbekannten Multicast-Gruppe angehören, kann dies effizient durch eine Multicast-Nachricht erfolgen. Jeder RepServer, welcher *DeleteGroup(G)* empfängt, löscht ebenfalls den Markencontainer zu *G* (siehe Abbildung 3.7). Um den Aufwand zu minimieren wird *DeleteGroup* mit unzuverlässigem IP-Multicast versendet. Dies ist hinreichend, da der Ablaufzeitmechanismus (siehe Abschnitt 3.6) für die zuverlässige Löschung der Marken sorgt.

---

*R* **empfängt** repDeleteGroup(*G*) **von** Klient *C*:
(1) lösche Markencontainer für Gruppe *G*;
(2) **sende** DeleteGroup(*G*) **an** Multicast-Gruppe der RepServer;

*R* **empfängt** DeleteGroup(*G*) **von** RepServer *S*:
(3) lösche Markencontainer für Gruppe *G*;

---

*Abbildung 3.7: Auflösen einer Gruppe in TRS-R*

#### 3.5.4.3 Gruppenbeitritt

Auch die *repJoinGroup(G,N,B)*-Operation wird, wie alle anderen Operationen auch, immer beim lokalen RepServer *R* ausgeführt. Daraufhin überprüft *R*, ob eine Marke der Gruppe *G* lokal vorhanden ist. Existiert eine lokale Marke, dann wird dieses aus dem Markenpaket durch Dekrementieren der Markenanzahl entfernt und an den neuen Teilnehmer *N* übergeben. Sind mehrere alternative Markenpakete vorhanden, dann wird ein Markenpaket mit dem kleinsten Metrikbetrag gewählt. Zusätzlich wird ein neues Markenpaket mit Eigentümer *N* und Anzahl *B* erzeugt, da *N* selbst Kindknoten im Kontrollbaum aufnehmen kann. Wird beispielsweise in Abbildung 3.4 (siehe Seite 42) angenommen, dass ein Knoten in der Domäne eins, $N_{12}$, eine *repJoinGroup*-Nachricht an seinen lokalen RepServer $S_1$ sendet, dann tritt die beschriebene Situation ein. $S_1$ hat eine lokale Marke der gewünschten Gruppe, welche an $N_{12}$ übergeben wird.

Gleichermaßen kann die Situation eintreten, in welcher der lokale RepServer *R* keine Marke der gewünschten Gruppe besitzt. Tritt dieser Fall ein, so startet *R* eine Multicast-basierte Markensuche. Die Suche beginnt mit einer Multicast *TokenSearch(G)*-Nachricht an die RepServer-Gruppenadresse mit einem Ausbreitungsbereich (TTL) von eins. *G* ist die gesuchte Gruppe in die der neue Teilnehmer eintreten möchte. Wenn nach einer angemessenen Wartezeit kein RepServer antwortet, dann wird die

```
R empfängt repJoinGroup(G, N, B) von Klient C:
(1)   if Markencontainer für G existiert then
(2)       TP := Markenpaket für Gruppe G mit lokal kleinstem Metrikbetrag;
(3)       TP.Anzahl := TP.Anzahl - 1;
(4)       sende Token an N;
(5)       if TP.Anzahl == 0 then
(6)           lösche TP;
(7)       erstelle neues Markenpaket für Gruppe G und Eigentümer N mit Anzahl B Marken;
(8)   else
(9)       sende TokenSearch(G) an Multicast-Gruppe mit TTL := 1;
(10)      case
(11)          | R empfängt TokenAvail(G, M) von RepServer P_1 ... P_x:
(12)              wähle P mit kleinstem Metrikbetrag M;
(13)              sende GetToken(G) an P;
(14)              case
(15)                  | R empfängt Token(Token) von RepServer P:
(16)                      sende Token(Token) an N;
(17)                      erstelle Markencontainer und Markenpaket für Gruppe G
(18)                          mit Eigentümer N und Anzahl B Marken;
(19)                  | R empfängt NoToken(G) von RepServer P:
(20)                      wähle anderen RepServer P aus P_1 ... P_x oder starte Suche in Zeile (9) erneut;
(21)          | Wartezeit abgelaufen:
(22)              if TTL < 255 then
(23)                  sende TokenSearch(G) an Multicast-Gruppe mit TTL := TTL+1
(24)                      und starte erneut in Zeile (10);
(25)              else
(26)                  keine Marke gefunden;
```

*Abbildung 3.8: Beitritt zu einer Gruppe in TRS-R*

Nachricht wiederholt gesendet mit einem jeweils erhöhten Ausbreitungsbereich. Dies wird durchgeführt, bis eine Antwort eintrifft oder der maximale Ausbreitungsbereich von 255 erreicht wird. Der Pseudocode in Abbildung 3.8 und 3.9 zeigt den kompletten Suchvorgang.

Ein RepServer *P*, der eine Suchnachricht erhält, überprüft, ob er eine gesuchte Marke besitzt. Ist dies gegeben, so antwortet er mit einer Unicast *TokenAvail(G,M)*-Nachricht. *M* ist der kleinste Metrikbetrag aller lokalen Marken. Eine gefundene Marke wird nicht direkt übergeben, da die Multicast-Suche mehrere Antworten zur Folge haben kann. Der suchende RepServer *R* ermittelt aus der Menge der Antworten die beste Marke aufgrund des übergebenen Metrikbetrags und sendet an *P* eine Unicast *GetToken*-Nachricht. Schließlich erhält *R* eine Marke von *P* mit einer Unicast *Token*-Nachricht und *P* löscht seine Marke. Nachdem *R* eine Marke erhalten hat und diese an den neuen Teilnehmer *N* übergeben hat, müssen wiederum neue Marken für den Eigentümer *N* angelegt werden. Diese werden bei *R* angelegt, da *N* zu der Domäne von *R* gehört. Nachfolgende Anmeldungen aus der selben Domäne können danach beträchtlich effizienter, ohne weitere Markensuche befriedigt werden.

Um dies an einem Beispiel aufzuzeigen, soll wieder Abbildung 3.4 dienen. Angenommen Knoten $N_{31}$ sendet eine *repJoinGroup*-Nachricht an seinen lokalen RepServer $S_3$. $S_3$ überprüft in dessen Fol-

```
R empfängt TokenSearch(G) von RepServer S:
(1)   if Markencontainer für G existiert then
(2)       TP := Markenpaket für Gruppe G mit kleinstem Metrikbetrag M;
(3)       sende TokenAvail(G, M) an S;

R empfängt GetToken(G) von RepServer S:
(4)   if Markencontainer für G existiert then
(5)       TP := Markenpaket für Gruppe G mit kleinstem Metrikbetrag M;
(6)       TP.Anzahl := TP.Anzahl - 1;
(7)       if TP.Anzahl == 0 then
(8)           lösche TP;
(9)       if TC.SP == ∅ des Markencontainers TC mit TC.GR == G then
(10)          lösche TC;
(11)      sende Token(Token) an S;
(12)  else
(13)      sende NoToken(G) an S;
```

*Abbildung 3.9: Markensuche in TRS-R*

ge, ob eine lokale Marke vorhanden ist und startet anschließend eine Markensuche. Die ersten drei Suchnachrichten treffen auf keinen RepServer, da sie zuvor von den Routern verworfen werden. Die nächste Suchnachricht mit einem Ausbreitungsbereich von vier wird von $S_2$ empfangen. Dieser besitzt ebenfalls keine Marken der gesuchten Gruppe. Erst die nächste Suchnachricht wird von $S_1$ empfangen, der eine passende Marke besitzt und deshalb mit einer *TokenAvail*-Nachricht antwortet. Der Empfang von *TokenAvail* durch $S_3$ führt zum Beenden der Markensuche. Durch eine *GetToken*– und einer anschließenden *Token*-Nachricht wird die eigentliche Markenübergabe durchgeführt. Die Marke wird schließlich von $S_3$ an $N_{31}$ übergeben und es werden neue Marken mit Eigentümer $N_{31}$ bei $S_3$ erzeugt. In Abbildung 3.10 ist die Markensuche durch ein Raum-Zeit-Diagramm dargestellt.

Während dieser zweiphasigen Interaktion — in der ersten Phase wird nach einer Marke gesucht und in der zweiten Phase eine Marke übergeben — kann es zu der folgenden Ausnahmesituation kommen. Wenn ein RepServer mit einer *TokenAvail*-Nachricht anzeigt, dass er für eine Markenübergabe bereit ist, dann wird diese Marke nicht bis zur eigentlichen Übergabe reserviert. Zum Beispiel kann ein RepServer, der nur eine Marke besitzt, dennoch auf mehrere Suchnachrichten antworten. Dies erlaubt die Verwendung eines einfachen und zustandslosen Protokolls. Da eine Marke nicht reserviert wird, muss die Reservierung auch nicht aufgehoben werden, wenn die Marke nicht übergeben wird. Dies ist immer dann der Fall, wenn mehrere RepServer auf eine Markensuche antworten. Dadurch kann andererseits die Situation auftreten, dass beim Empfang der *GetToken*-Nachricht ein RepServer keine Marke der gesuchten Gruppe mehr besitzt. In dessen Folge antwortet der RepServer mit einer *NoToken*-Nachricht. Der suchende RepServer wählt anschließend einen anderen RepServer aus, da evtl. mehrere *TokenAvail*-Nachrichten eingegangen sind. Ansonsten wird die Markensuche mit weiteren Suchnachrichten und einem erhöhten Ausbreitungsbereich fortgesetzt.

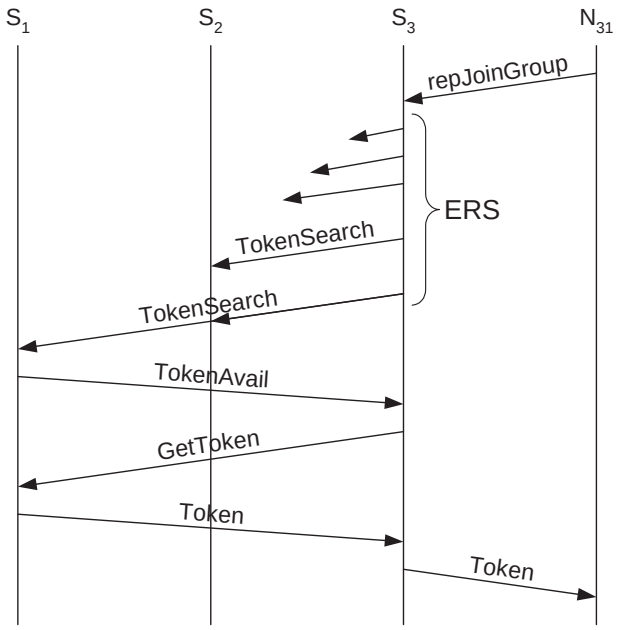



*Abbildung 3.10: Markensuche mittels Multicast in TRS-R*

#### 3.5.4.4 Gruppenaustritt

Verlässt ein Teilnehmer *N* den Kontrollbaum, so müssen alle Marken, für die er der Eigentümer ist, gelöscht werden. Das verwendete Multicast-Transportprotokoll ist verantwortlich dafür, dass ein Teilnehmer nur dann den Kontrollbaum verlassen darf, wenn er selbst keine Kindknoten mehr besitzt, d.h. ein Blattknoten im Kontrollbaum ist. Ein Teilnehmer mit Kindknoten muss vor dem Verlassen des Kontrollbaums dafür Sorge tragen, dass alle Kindknoten sich an einer anderen Stelle im Kontrollbaum anmelden. Alle Marken mit Eigentümer *M* befinden sich beim lokalen RepServer *R* von *M*. Wenn *R* die *repLeaveGroup(G,M)*-Nachricht empfängt, wird das Markenpaket mit Gruppe *G* und Eigentümer *M* aus dem Repository entfernt. Der Gruppenaustritt von *M* betrifft zudem nicht nur die Marken von *M*, sondern auch den Gruppenführer von *M* im Kontrollbaum, da nun wieder für einen Kindknoten ein Platz frei wird. Deshalb muss der Gruppenführer von *M* wieder eine Marke dem Repository hinzufügen. Von *M* bzw. seinem lokalen RepServer *R* kann dies nicht durchgeführt werden, da der lokale RepServer des Gruppenführers nicht bekannt ist. Daher führt der Gruppenführer an seinem lokalen RepServer eine *repAddToken*-Operation durch. Abbildung 3.11 zeigt den Pseudocode der dazugehörenden Operationen.

Zu beachten ist zudem, dass diese Vorgehensweise für zusätzliche Robustheit sorgt. Unter der Annahme, dass ein Knoten ausfallen kann wird deutlich, dass der ausgefallene Knoten selbst keine Marke hinzufügen kann. Dies kann lediglich vom Gruppenführer durchgeführt werden, nachdem er den Ausfall seines Kindknotens bemerkt hat. Vollständige Transportprotokolle haben üblicherweise Me-

chanismen vorgesehen, um den Ausfall von Kindknoten durch fehlende Lebenszeichennachrichten zu detektieren und den Knoten aus dem Kontrollbaum zu entfernen [Chiu et al. 1998].

```
R empfängt repLeaveGroup(G, M) von Klient C:
(1)   lösche Markenpaket für Gruppe G und Eigentümer M;
(2)   if Markencontainer für Gruppe G == ∅ then
(3)       lösche Markencontainer;

R empfängt repAddToken(G, E) von Klient C:
(4)   TP := Markenpaket für Gruppe G mit Eigentümer E;
(5)   TP.Anzahl := TP.Anzahl + 1;
```

*Abbildung 3.11: Verlassen einer Gruppe in TRS-R*

In Tabelle 3.1 sind alle internen Nachrichten des TRS-R-Diensts nochmals zusammengefasst.

### 3.5.5 Strategie 2: TRS mit minimalen Kosten (TRS-K)

Die Domäneneinteilung des Netzwerks wird bei TRS-K und dem anschließend besprochenen TRS-M erweitert. Neben der disjunkten Zuordnung jedes Knotens zu einer Domäne, werden zudem die Domänen selbst hierarchisch strukturiert. Jeder Knoten des Netzwerks ist einer Blattdomäne zugeordnet. Domänen können zu einer übergeordneten Domäne zusammengefasst werden bis schließlich eine Hierarchie entsteht, genannt *TRS-Baum*, in der die Wurzeldomäne alle anderen Domänen umfasst. Abbildung 3.12 zeigt dies anhand eines Beispiels. RepServer $S_1$ ist für die Blattdomäne 1 verantwortlich und RepServer $S_4$ für die Domäne 4, die aus mehreren Unterdomänen besteht. Ein Klient greift auf den TRS-Dienst generell über die Blatt-RepServer zu und zwar jeweils über den Blatt-RepServer seiner Domäne. Die gesamte Markeninformation ist bei den Blatt-RepServern gespeichert. Die verbleibenden RepServer sind zur effizienten Markensuche notwendig. Wie in Abschnitt 3.5.1 erläutert wurde entlastet dies die Such-RepServer.

| Nachricht | Typ | Beschreibung |
|---|---|---|
| DeleteGroup(G) | Multicast | Löschen der Gruppe *G* im TRS, d.h. löschen aller Markencontainer für Gruppe *G*. |
| TokenSearch(G) | Multicast | Suchen einer Marke der Gruppe *G*. |
| TokenAvail(G, M) | Unicast | Antwort auf TokenSearch, um die Verfügbarkeit einer Marke mit Metrik *M* zu signalisieren. |
| GetToken(G) | Unicast | Antwort auf TokenAvail zur Anforderung der Markenübergabe. |
| Token(Token) | Unicast | Übergabe einer Marke als Antwort auf GetToken. |
| NoToken(G) | Unicast | Antwort auf GetToken, wenn keine Marke verfügbar ist. |

*Tabelle 3.1: Interne Nachrichten von TRS-R*

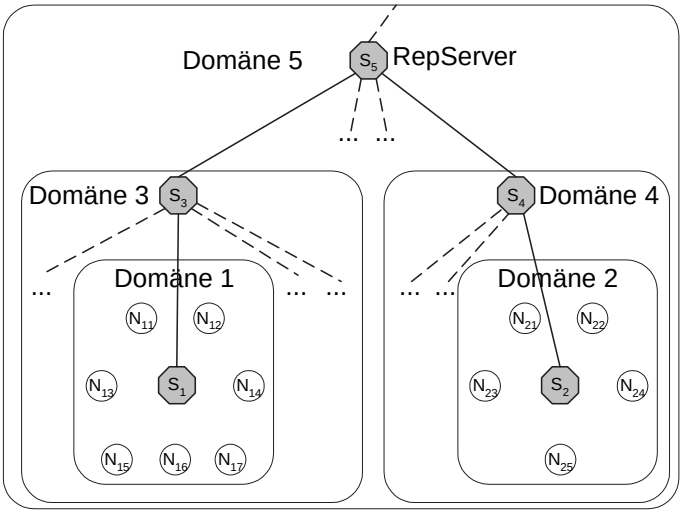



*Abbildung 3.12: Hierarchische Domäneneinteilung*

Der *Gruppenbaum* ist ein Teilbaum des TRS-Baums und existiert für jede bekannte Multicast-Gruppe. Der Gruppenbaum einer Gruppe besteht aus allen Blatt-RepServern im TRS-Baum, die Marken dieser Gruppe speichern und allen transitiven Vaterknoten im TRS-Baum, d.h. der Wurzelknoten des TRS-Baums ist auch die Wurzel jedes Gruppenbaums. Die Zugehörigkeit zum Gruppenbaum wird in den daran beteiligten Such-RepServern in einem gruppenspezifischen *Gruppenregister* gespeichert. Das Gruppenregister beinhaltet folgende drei Datensätze: (1) die Gruppe *GR* des Gruppenregisters, (2) die Teilbäume *TB* und (3) die Ablaufzeit *AZ*. *TB* ist ein Feld der Länge *K*, bestehend aus je einem Bit, wobei *K* den maximalen Verzweigungsgrad des TRS-Baums bezeichnet. Jedes Bit korrespondiert mit einem Kindknoten im TRS-Baum. Ist das Bit für einen Kindknoten gesetzt, gehört dieser dem Gruppenbaum an. Die Notation *TB[i]* soll den Eintrag des *TB*-Vektors für den Teilbaum *i* kennzeichnen. Die Ablaufzeit der Gruppenregister wurde wie die Ablaufzeit der Marken aus Gründen der Robustheit in Fehlersituationen integriert. Eine nähere Beschreibung erfolgt in Abschnitt 3.6.

Der Gruppenbaum und folglich auch das Gruppenregister ist nicht statisch, da mit dem Eintritt und Austritt von Multicast-Teilnehmern Marken erzeugt respektive gelöscht werden. Die Aktualisierung des Gruppenregisters wird im folgenden Abschnitt generisch beschrieben, bevor die Gruppenoperationen und ihre Auswirkung auf den Gruppenbaum bzw. ihre Verwendung des Gruppenbaums im Speziellen dargelegt werden.

#### 3.5.5.1 Aktualisierung des Gruppenregisters

Ein Gruppenbaum dehnt sich während der Lebenszeit der zugehörigen Multicast-Gruppe aus, der gesamte TRS-Baum kann dann zum Gruppenbaum gehören, kann anschließend wieder schrumpfen,

wenn das Interesse an einer Gruppe nachlässt und dann evtl. wieder einen neuen Ausdehnungszyklus beginnen. Nachdem ein Markencontainer angelegt wurde, muss der betreffende Blatt-RepServer in den Gruppenbaum eingegliedert werden. Dazu wird eine *TokenAvail*-Nachricht (siehe Tabelle 3.2 auf Seite 61) an den Vaterknoten im TRS-Baum gesendet. Diese Nachricht wird Schritt für Schritt in Wurzelrichtung weitergeleitet, bis sie von einem RepServer empfangen wird, der bereits zum Gruppenbaum dieser Gruppe gehört oder der Wurzel-RepServer erreicht ist. Alle außer dem Blatt-RepServer legen ein Gruppenregister an. Im *TB*-Feld des Gruppenregisters wird das entsprechende Bit des sendenden Kindknotens auf eins gesetzt und damit dessen Zugehörigkeit zum Gruppenbaum dokumentiert.

Wird ein Markencontainer auf einem Blatt-RepServer gelöscht, dann muss der RepServer aus dem Gruppenbaum ausgegliedert werden, da er keine Marken mehr bereitstellen kann. Dazu sendet der Blatt-RepServer eine *NoTokenAvail*-Nachricht an dessen Vaterknoten im TRS-Baum. Auch diese Nachricht wird Schritt für Schritt in Wurzelrichtung weitergeleitet, bis ein RepServer erreicht wird, der außer dem sendenden Knoten noch weitere Kindknoten in seinem Gruppenregister besitzt oder wiederum der Wurzelknoten erreicht ist. Generell setzt ein Empfänger den Sendeknoten im *TB*-Feld des Gruppenregisters auf null. Damit wird die sendende Teilhierarchie aus dem Gruppenbaum entfernt. War dies die einzige Teilhierarchie im Gruppenbaum, so wird das Gruppenregister komplett gelöscht. In Abbildung 3.13 ist der gesamte Ablauf in Pseudocode dargestellt.

```
R empfängt TokenAvail(G) von Kindknoten S:
(1)   Reg := Gruppenregister mit Reg.GR == G;
(2)   if not exists Reg then
(3)       erstelle Gruppenregister Reg für Gruppe G;
(4)       sende TokenAvail(G) an Vaterknoten;
(5)   Reg.TB[S] := 1;

R empfängt NoTokenAvail(G) von Kindknoten S:
(6)   Reg := Gruppenregister mit Reg.GR == G;
(7)   Reg.TB[S] := 0;
(8)   if ∀ i : Reg.TB[i] == 0 then
(9)       lösche Reg;
(10)      sende NoTokenAvail(G) an Vaterknoten;
```

*Abbildung 3.13: Aktualisieren der Gruppenregister in TRS-K*

#### 3.5.5.2 Gruppenerstellung

Das Erzeugen einer Gruppe bedingt das Anlegen eines initialen Gruppenbaums. Angenommen, *R* sei der Blatt-RepServer, an dem die *repCreateGroup(G,N,B)*-Operation durchgeführt wird, dann besteht der Gruppenbaum aus *R* und allen transitiven Vaterknoten im TRS-Baum bis zur Wurzel. Zuerst

wird *R* einen Markencontainer für Gruppe *G* mit einem Markenpaket für Eigentümer *N* und Anzahl *B* anlegen. Anschließend sendet *R* eine *TokenAvail*-Nachricht an seinen Vaterknoten, der das Gruppenregister zur Markensuche anlegt und die Nachricht weiter in Richtung Wurzel sendet. Im vorigen Abschnitt zur Aktualisierung der Gruppenregister wurde dies bereits im Detail erläutert. In Abbildung 3.14 ist der Vorgang in Pseudocode dargestellt.

Abbildung 3.15 soll den Vorgang beispielhaft verdeutlichen. Abbildung 3.15a zeigt den TRS-Baum bestehend aus sechs RepServern. Ein Klient $N_{11}$, der sich in den Kontrollbaum eingliedern möchte, sendet eine *repCreateGroup*-Nachricht an seinen lokalen RepServer $S_1$. In Abbildung 3.15b sind die Marken und die Gruppenregister nach dem erfolgreichen Anlegen des initialen Gruppenbaums dargestellt. Wiederum wird als Metrik die Höhe des Markeneigentümers im Kontrollbaum gewählt, die für die Wurzel des Kontrollbaums eins beträgt.

*R* **empfängt** repCreateGroup(*G*, *N*, *B*) **von** Klient *C*:
(1) erstelle Markenpaket mit *B* Marken für Eigentümer *N* und belege den initialen Metrikbetrag;
(2) erstelle Markencontainer für Gruppe *G* und füge das Markenpaket ein;
(3) **sende** TokenAvail(*G*) **an** Vaterknoten; // siehe Abbildung 3.13

*Abbildung 3.14: Erstellen einer Gruppe in TRS-K*

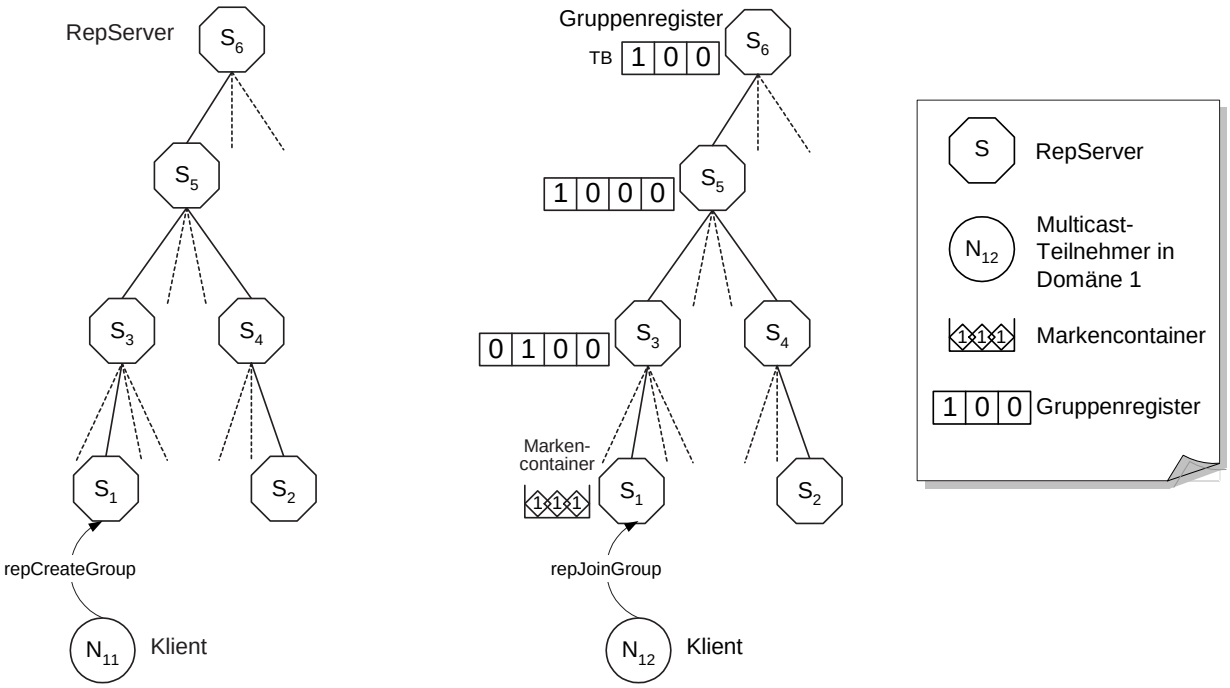



*Abbildung 3.15: (a) Erstellung einer Gruppe und (b) Beitritt zu einer Gruppe in TRS-K*

#### 3.5.5.3 Gruppenauflösung

Der zur Löschung einer Gruppe beauftragte RepServer löscht den Markencontainer der betreffenden Gruppe und sendet eine *DeleteGroup*-Nachricht an seinen Vaterknoten im TRS-Baum. Jeder Such-RepServer bis auf die Wurzel leitet die *DeleteGroup*-Nachricht Schritt für Schritt in Richtung Wurzelknoten weiter. Ist ein Such-RepServer selbst Mitglied des Gruppenbaums, so wird die Nachricht auch an die Kindknoten weitergeleitet, die zum Gruppenbaum gehören (siehe Abbildung 3.16). Jeder RepServer löscht beim Empfang der Nachricht sein Gruppenregister bzw. seinen Markencontainer der betreffenden Gruppe. Zu beachten ist, dass dies nicht durch zuverlässige Kommunikation erfolgen muss, da die Ablaufzeit im Fehlerfall für die Löschung der Informationen sorgt.

```
R empfängt repDeleteGroup(G) von Klient C:
(1)   lösche Markencontainer für Gruppe G;
(2)   sende DeleteGroup(G) an Vaterknoten;

R empfängt DeleteGroup(G) von Kindknoten S:
(3)   if R is not Wurzelknoten then
(4)       sende DeleteGroup(G) an Vaterknoten;
(5)   Reg := Gruppenregister mit Reg.GR == G;
(6)   forall i with Reg.TB[i] == 1 außer S do
(7)       sende DeleteGroup(G) an Kindknoten i;
(8)   lösche Reg;

R empfängt DeleteGroup(G) von Vaterknoten S:
(9)   if R is not Blattknoten then
(10)      Reg := Gruppenregister mit Reg.GR == G;
(11)      forall i with Reg.TB[i] == 1 do
(12)          sende DeleteGroup(G) an Kindknoten i;
(12)      lösche Reg;
(14)  else
(15)      lösche Markencontainer für Gruppe G;
```

*Abbildung 3.16: Auflösen einer Gruppe in TRS-K*

#### 3.5.5.4 Gruppenbeitritt

Der Eintritt in den Kontrollbaum wird von einem neuen Teilnehmer *N* mit der Nachricht *repJoinGroup(G,N,B)* an seinen lokalen RepServer *R* angestoßen. Daraufhin prüft *R*, ob eine Marke für die gesuchte Gruppe *G* lokal vorhanden ist. Ist dem so, dann wird eine Marke aus dem Markenpaket durch Dekrementieren der Markenanzahl entnommen und an *N* übergeben. Stehen mehrere alternative Markenpakete zur Auswahl, dann wird das Markenpaket mit dem kleinsten Metrikbetrag gewählt. Schließlich wird noch ein neues Markenpaket mit Eigentümer *N* und Markenanzahl *B* erzeugt und im Markencontainer abgelegt. In Abbildung 3.15b ist der Klient $N_{12}$ in der selben Domäne wie $N_{11}$ und

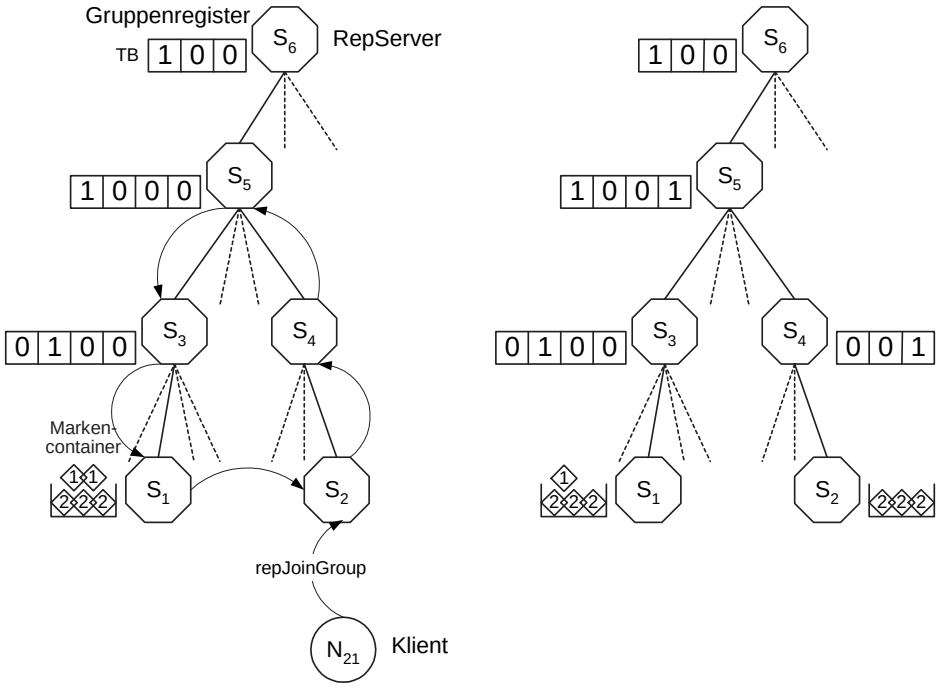



*Abbildung 3.17: (a) Beitritt zu einer Gruppe und (b) Gruppenregister nach dem Beitritt in TRS-K*

sendet darum seine *repJoinGroup*-Nachricht ebenfalls an $S_1$. Da $S_1$ aufgrund der Gruppenerstellung von $N_{11}$ Marken der Gruppe besitzt, kann die Anfrage lokal befriedigt werden. Abbildung 3.17a zeigt die Situation des Markencontainers nach der Anmeldung. Bisher ergibt sich noch kein Unterschied zu TRS-R.

Stehen andererseits keine lokalen Marken zur Verfügung, so startet *R* eine Markensuche über den TRS-Baum. Die Markensuche besteht aus zwei Phasen: (1) der Phase der Suchraumausdehnung vom Blatt-RepServer in Wurzelrichtung und (2) der Phase der Suchraumeinschränkung von einem Such-RepServer in Richtung Blatt-RepServer. *R* initiiert die erste Suchphase, namentlich die Suchraumausdehnung, indem eine *TokenSearch*-Nachricht an den Vaterknoten im TRS-Baum gesendet wird. Diese Nachricht wird Schritt für Schritt in Wurzelrichtung weitergeleitet, bis entweder ein RepServer gefunden wird, der zum Gruppenbaum von *G* gehört oder die Wurzel erreicht wird. Der gefundene RepServer kehrt nun die Suchrichtung in Richtung Blatt-RepServer um und schränkt damit den Suchraum immer stärker ein, bis schließlich ein Blatt-RepServer mit einer Marke gefunden wird. Die *TokenSearch*-Nachricht wird dazu von jedem beteiligten Such-RepServer jeweils an einen Kindknoten gesendet, dessen zugehöriger Eintrag im Teilbaum des Gruppenregisters gesetzt ist und damit die Existenz einer Marke für diesen Teilbaum verspricht. Dies wird von jedem Such-RepServer Schritt für Schritt wiederholt. Erfüllen mehrere Teilbäume im Gruppenregister diese Anforderung, so wird ein Teilbaum zufällig ausgewählt. Zu beachten ist, dass die TRS-K Strategie keine Möglichkeit bietet, bei der Markensuche Metriken zu berücksichtigen. Diese Möglichkeit wird erst mit der TRS-M

Strategie eingeführt werden.

```
R empfängt repJoinGroup(G, N, B) von Klient C:
(1)   TP := Markenpaket für Gruppe G mit lokal kleinstem Metrikbetrag;
(2)   if TP exists then
(3)       TP.Anzahl := TP.Anzahl - 1;
(4)       sende Token an N;
(5)       if TP.Anzahl == 0 then
(6)           lösche TP;
(7)       erstelle neues Markenpaket für Gruppe G und Eigentümer N mit Anzahl B Marken;
(8)   else
(9)       sende TokenSearch(G, N, B, R) an Vaterknoten;

R empfängt TokenSearch(G, N, B, L) von Kindknoten S:
(10)  Reg := Gruppenregister für Gruppe G;
(11)  if Reg exists then
(12)      sende TokenSearch(G, N, B, L) an Kindknoten i mit Reg.TB[i] == 1;
(13)  else
(14)      sende TokenSearch(G, N, B, L) an Vaterknoten;

R empfängt TokenSearch(G, N, B, L) von Vaterknoten S:
(15)  if R != Blatt-RepServer then
(16)      Reg := Gruppenregister für Gruppe G;
(17)      sende TokenSearch(G, N, B, L) an Kindknoten i mit Reg.TB[i] == 1;
(18)  else
(19)      TP := Markenpaket für Gruppe G mit lokal kleinstem Metrikbetrag;
(20)      TP.Anzahl := TP.Anzahl - 1;
(21)      sende Token(G, N, B, TP.M) an L;
(22)      if TP.Anzahl == 0 then
(23)          lösche TP;
(24)      if Markencontainer TC für Gruppe G mit TC.SP == ∅ then
(25)          lösche TC;
(26)          sende NoTokenAvail(G) an Vaterknoten;

R empfängt Token(G, N, B, M) von RepServer S:
(27)  sende Token an N;
(28)  erstelle Markencontainer und Markenpaket
          für Gruppe G mit Eigentümer N und Anzahl B Marken;
(29)  sende TokenAvail(G) an Vaterknoten;
```

*Abbildung 3.18: Beitritt zu einer Gruppe in TRS-K*

Der gefundene Blatt-RepServer entnimmt eine Marke mit kleinstem Metrikbetrag dem Markencontainer für Gruppe $G$ und liefert es direkt an den suchenden RepServer $R$. Die Übergabe der Marke unter erneuter Zuhilfenahme des TRS-Baums wäre alternativ denkbar, zusammen mit einer gegebenenfalls notwendigen Aktualisierung der Gruppenregister, verursacht aber höhere Kommunikationskosten. Nach dem Empfang der Marke und der Übergabe an $N$, legt $R$ einen Markencontainer für $G$ mit einem Markenpaket für Eigentümer $N$ und Markenanzahl $B$ an (siehe Code in Abbildung 3.18). Schließlich verbleibt noch die Aufgabe, $R$ mit dem Gruppenbaum zu verbinden, da bei einer potenti-

ell folgenden Markensuche auch *R* Marken liefern kann. RepServer *R* sendet zu diesem Zweck eine *TokenAvail*-Nachricht an seinen Vaterknoten (siehe Abschnitt 3.5.5.1).

Ein Beispiel soll dies verdeutlichen. Angenommen, Klient $N_{21}$ in Abbildung 3.17a möchte sich mit dem Kontrollbaum verbinden, dann kontaktiert er seinen lokalen RepServer $S_2$ mit einer *repJoinGroup*-Nachricht. Da $S_2$ keine lokalen Marken besitzt, wird eine Markensuche gestartet. In der ersten Suchphase wird der Suchraum ausgedehnt, indem die Suche jeweils an den Vaterknoten weitergereicht wird. Die erste Suchphase endet bei RepServer $S_5$, der Mitglied des Gruppenbaums ist. Das Gruppenregister wird nun dazu verwendet, in der zweiten Suchphase den Blatt-RepServer $S_1$ mit Marken zu finden. $S_1$ entnimmt eine Marke seinem Markencontainer und liefert es direkt an $S_2$. $S_2$ sendet daraufhin eine *TokenAvail*-Nachricht an seinen Vaterknoten $S_4$. $S_4$ legt ein Gruppenregister an und setzt die Teilbaumhierarchie von $S_2$ auf eins, so dass eine spätere Markensuche auch $S_2$ finden kann. Die *TokenAvail*-Nachricht wird schließlich an $S_5$ weitergereicht, der die Verbindung zum Gruppenbaum herstellt. Abbildung 3.17b zeigt die Gruppenregister nach der Aktualisierung.

Der beschriebene Algorithmus erlaubt, dass Anmeldungen vom lokalen RepServer möglichst selbst durchgeführt werden können. Ist dies aufgrund fehlender lokaler Marken nicht erreichbar, wird eine Markensuche gestartet, die allerdings immer einen möglichst kleinen Bereich durchsucht, deren Gruppenführer bezüglich der Abstände im Netzwerk in der Nähe des Klienten liegen.

#### 3.5.5.5 Gruppenaustritt

Wenn ein Mitglied *M* den Kontrollbaum verlässt wird eine Marke freigegeben, die den Gruppenführer von *M* im Kontrollbaum als Eigentümer hat. Folglich muss der Gruppenführer mit der *repAddToken*-Operation bei seinem lokalen RepServer wieder eine Marke einfügen. Das Markenpaket mit Eigentümer *M* wird durch eine *repLeaveGroup*-Nachricht an den lokalen RepServer von *M* gelöscht. Wenn dadurch der gesamte Markencontainer leer wird, sendet der RepServer eine *NoTokenAvail*-Nachricht an den Vaterknoten im Kontrollbaum, die gegebenenfalls laut der Beschreibung in Abschnitt 3.5.5.1 Schritt für Schritt in Richtung Wurzel weitergeleitet wird, um den Gruppenbaum an die neue Situation anzupassen. Der gesamte Algorithmus ist in Abbildung 3.19 dargestellt.

Zusammenfassend finden sich in Tabelle 3.2 alle internen Operationen von TRS-K.

### 3.5.6 Strategie 3: TRS mit Metriken (TRS-M)

Die Beschreibung der Protokolle TRS-R und TRS-K wies bereits auf eine Verbesserungsmöglichkeit hin. Zwar kann innerhalb eines RepServers die beste Marke bezüglich der Metrik ermittelt und an einen neuen Teilnehmer übergeben werden, allerdings bietet weder TRS-R noch TRS-K eine Möglichkeit bei der globalen Markensuche die Metrik zu berücksichtigen. In TRS-R könnte dies durch

| Nachricht | Typ | Beschreibung |
|---|---|---|
| TokenAvail(G) | Unicast | Einrichten eines Gruppenregistereintrags. |
| NoTokenAvail(G) | Unicast | Löschen eines Gruppenregistereintrags. |
| DeleteGroup(G) | Unicast | Löschen der Gruppe *G* |
| TokenSearch(G, N, B, L) | Unicast | Suchen einer Marke der Gruppe *G*, ausgehend von RepServer *L*. Der neue Teilnehmer *N* hat einen maximalen Verzweigungsgrad von *B*. |
| Token(G, N, B, M) | Unicast | Übergabe einer Marke mit Metrikbetrag *M*; *N* und *B* beziehen sich auf den neuen Teilnehmer (siehe oben). |

*Tabelle 3.2: Interne Nachrichten von TRS-K*

*R* **empfängt** repLeaveGroup(*G*, *M*) **von** Klient *C*:
(1) lösche Markenpaket für Gruppe *G* und Eigentümer *M*;
(2) **if** Markencontainer *TC* für Gruppe *G* mit *TC.SP* == ∅ **then**
(3) lösche *TC*;
(4) **sende** NoTokenAvail(*G*) **an** Vaterknoten;

*R* **empfängt** repAddToken(*G*, *E*) **von** Klient *C*:
(5) *TP* **:=** Markenpaket für Gruppe *G* mit Eigentümer *E*;
(6) *TP.Anzahl* **:=** *TP.Anzahl* + 1;

*Abbildung 3.19: Verlassen einer Gruppe in TRS-K*

eine globale Suche mittels unbegrenztem Multicast und anschließender Auswahl der besten Antwort realisiert werden. Allerdings ist dies kein skalierbarer Ansatz, da die Netz– und Knotenbelastung durch die globale Multicast-Suche und die vielen Antwortnachrichten enorm wäre. Eine Suche unter Einbeziehung aller RepServer im TRS-Baum des TRS-K Ansatzes scheidet aus den gleichen Gründen aus. Ein vielverwendeter Ansatz solch eine Suche zu realisieren, wird durch allgemeine Suchalgorithmen beschrieben [Sedgewick 1991]. Eine hierarchische Strukturierung des Suchraums erlaubt die effiziente Suche nach einem Schlüssel (hier Metrikbetrag). Ein Beispiel eines solchen Algorithmus stellen B-Bäume dar [Comer 1979, Sedgewick 1991, S. 308]. Mit dem TRS-Baum ist bereits eine hierarchische Struktur vorhanden, die sich dafür eignet. Die Gruppenregister müssen dazu erweitert werden, um über die eingeschränkte Angabe, ob eine Marke in einem Teilbaum gefunden werden kann, auch den kleinsten Metrikbetrag des Teilbaums bereitzustellen.

Das Gruppenregister von TRS-M enthält nun folgende Daten: (1) die Gruppe *GR* des Gruppenregisters, (2) den kleinsten Metrikbetrag im gesamten Gruppenbaum *GM*, (3) den Vektor *TBM* mit dem kleinsten Metrikbetrag für jeden Teilbaum und (4) die Ablaufzeit *AZ*. Auch auf den Blatt-RepServern wird fortan ein vereinfachtes Gruppenregister gespeichert, das folgende Daten enthält: (1) die Gruppe *GR* des Gruppenregisters, (2) den kleinsten Metrikbetrag im gesamten Gruppenbaum *GM*, (3) den kleinsten Metrikbetrag aller lokaler Marken *LM* und (4) die Ablaufzeit *AZ*. Für den Vektor *TBM* wird

ähnlich zu TRS-K die Notation *TBM[i]* eingeführt, um auf den Metrikbetrag des Teilbaums *i* zuzugreifen. Gehört ein Teilbaum *i* nicht dem Gruppenbaum an, so ist *TBM[i]* unbelegt, was durch das Symbol $\perp$ gekennzeichnet wird. Die formale Definition der Metriksangaben lautet:

$$\begin{aligned} LM &= \forall\, T \in \{lokale\, Markenpakete\} : min\{T.Metrikbetrag\} \\ TBM[i] &= \begin{cases} \forall\, j \in \{i.TBM[j]\} : min\{j.TBM\} & , i \text{ ist Such-RepServer} \\ i.LM & , i \text{ ist Blatt-RepServer} \end{cases} \\ GM &= \forall i : min\{Wurzel\text{-}RepServer.TBM[i]\}. \end{aligned}$$

Während die Notwendigkeit von *TBM* offensichtlich gegeben ist, da sonst eine Suche im TRS-Baum nach der besten Marke nicht möglich ist, muss die Notwendigkeit von *GM* noch begründet werden. Mit *TBM* alleine wäre es möglich, startend von der TRS-Wurzel, nach der besten Marke zu suchen. Eine derartige Belastung der Wurzel für jede Eingliederung in den Kontrollbaum wäre aber kaum tragbar. Die Suche soll weiterhin von den Blatt-RepServern ausgehen und möglichst lokal beschränkt, d.h. nur in seltenen Fällen bis zur Wurzel weitergereicht werden. Dies erfordert bei jedem RepServer die Kenntnis darüber, ob es lohnenswert ist, den Suchbereich weiter auszudehnen, d.h. den Vaterknoten mit einzubeziehen oder ob bereits aufgrund lokaler Information gesichert bestimmt werden kann, dass bessere Marken durch die Suchraumausdehnung nicht gefunden werden können. Genau diese Information wird durch *GM* zur Verfügung gestellt. Besitzt der Teilbaum eines RepServers bereits Marken mit dem Metrikbetrag *GM*, also wenn $min(\forall\ i\ \ TBM[i]) = GM$ zutrifft, so kann auf eine weitere Suchraumausdehnung verzichtet werden. Im folgenden Abschnitt soll dargelegt werden, wie *TBM* und *GM* verwaltet werden.

#### 3.5.6.1 Aktualisierung des Gruppenregisters

Neben dem reinen Erweitern und Kürzen des Gruppenbaums, d.h. Eingliedern und Entfernen von RepServern, müssen die Metrikbeträge für die Teilbäume und die globale Metrik angepasst werden. Dazu wird die Nachricht *Update*, ausgehend von den Blatt-RepServern, an den Vaterknoten im TRS-Baum gesendet, wenn sich lokal eine Veränderung dieser Metrikbeträge ergeben hat. In Abbildung 3.20 ist der vollständige Algorithmus dargestellt.

Der globale Metrikbetrag *GM* wird immer dann angepasst, wenn eine durchgeführte globale Markensuche im Gruppenbaum keine Marke des bisherigen Metrikbetrags von *GM* gefunden hat. Zu beachten ist, dass *GM* nicht in allen Gruppenregistern den aktuellen globalen Metrikbetrag widerspiegelt, da aus Gründen des geringeren Nachrichtenaufwands neue Werte nur in Richtung der Wurzel aber nicht von dort aus in Richtung aller Blattknoten propagiert werden. Daher ist *GM* nur eine untere Schranke des globalen Metrikbetrags. Der vorgestellte Algorithmus arbeitet bereits korrekt mit der Kenntnis der unteren Schranke. Gegenüber dem globalen Propagieren neuer Metrikbeträge über den gesamten

```
R empfängt Update(G, TBM, GM) von Kindknoten S:
(1)   Reg := Gruppenregister mit Reg.GR == G;
(2)   if Reg existiert then
(3)       Reg.TBM[S] := TBM;
(4)       if GM > Reg.GM then
(5)           Reg.GM := GM;
(6)       if (min(Reg.TBM) or Reg.GM has changed) and (R not Wurzelknoten) then
(7)           send Update(G, min(Reg.TBM), Reg.GM) an Vaterknoten;
(8)       if (∀ i Reg.TBM[i] == ⊥) then
(9)           lösche Reg;
(10)  elseif TBM != ⊥ then
(11)      lege Reg mit Reg.GR := G und Reg.TBM[S] := TBM und Reg.GM := GM an;
(12)      if R not Wurzelknoten then
(13)          send Update(G, min(Reg.TBM), Reg.GM) an Vaterknoten;
```

*Abbildung 3.20: Aktualisierung des Gruppenbaums in TRS-M*

TRS-Baum spart dies nicht nur Kosten, sondern befreit den Algorithmus auch von der Beachtung möglicher nebenläufigkeitsbedingter Konsistenzprobleme, was schließlich leichtgewichtigere Protokolle ermöglicht.

#### 3.5.6.2 Gruppenerstellung und Gruppenauflösung

Bei der Gruppenerstellung muss analog zu TRS-K der initiale Gruppenbaum vom Blatt-RepServer bis zum Wurzel-RepServer etabliert werden (siehe Algorithmus in Abbildung 3.13). Dazu wird die Nachricht *Update* Schritt für Schritt an den Vaterknoten weitergeleitet. Der initiale Metrikbetrag der ersten Marken wird gleichermaßen für die Belegung von *GM* als auch *TBM* der Gruppenregister herangezogen. Das Auflösen einer Gruppe ist identisch zu TRS-K (siehe Algorithmus in Abbildung 3.16).

In Abbildung 3.21 ist die Situation der Gruppenerstellung beispielhaft dargestellt. Ein neuer Teilnehmer $N_{11}$ sendet *repCreateGroup* an seinen lokalen RepServer $S_1$. Zusätzlich zum Anlegen des Markencontainers wird der initiale Gruppenbaum erstellt. Als Metrikbetrag wird durchgängig eins eingetragen, die Metrik der initialen Marken als Höhe des Markeneigentümers im Kontrollbaum.

#### 3.5.6.3 Gruppenbeitritt

Beim Gruppenbeitritt muss ein aufgerufener Blatt-RepServer *R* nicht nur für die Lieferung einer beliebigen Marke Sorge tragen, sondern muss eine Marke mit kleinstem Metrikbetrag, bzw. eine mit definierter maximalen Abweichung zum kleinsten Metrikbetrag, liefern. Eine derartige Marke ist lokal vorhanden, wenn $LM \leq GM + Toleranz$ ist. *Toleranz* ist ein wählbarer nichtnegativer Parameter,

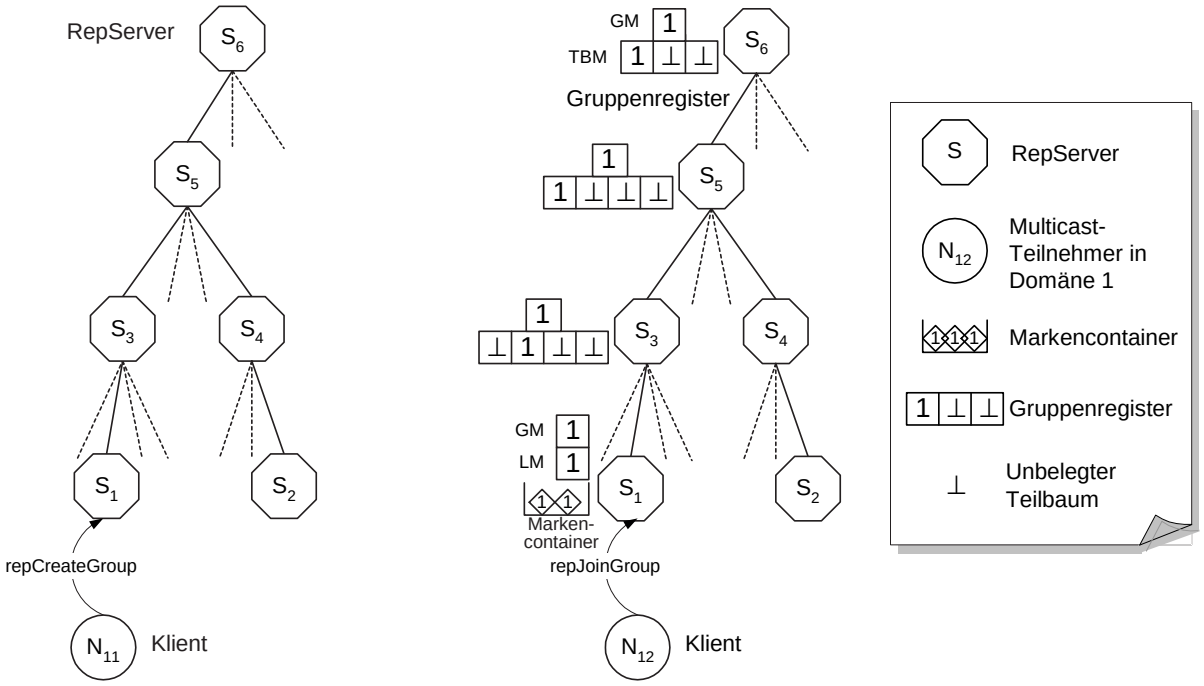



*Abbildung 3.21: (a) Erstellen einer Gruppe und (b) Beitritt zu einer Gruppe in TRS-M*

mit dem die Suche nach dem kleinsten Metrikbetrag abschwächbar ist. Damit kann die Markensuche mit geringerem Nachrichtenaufwand durchgeführt werden. Eine *Toleranz* von null resultiert in der Rückgabe einer Marke mit global kleinstem Metrikbetrag dieser Gruppe, während eine *Toleranz* von eins eine Abweichung des Metrikbetrags um eins vom kleinsten Betrag zulässt und für andere Toleranzen entsprechend. Eine Marke ist eine *ideale Marke*, wenn ihr Metrikbetrag $M_1$ folgende Bedingung erfüllt: $\forall$ Marken $t$ einer gegebenen Multicast-Gruppe : $M_1 \leq t.M + Toleranz$, d.h. wenn es keine andere Marke im TRS-Dienst gibt, die einen kleineren Metrikbetrag aufweist, es sei denn die *Toleranz* gestattet eine definierte Abweichung.

Wenn der lokale RepServer *R* eine ideale Marke besitzt, so wird diese an den neuen Teilnehmer übergeben. Der Pseudocode in Abbildung 3.22 beschreibt diesen Ablauf. Nach der Lieferung einer Marke muss eventuell *LM* angepasst werden, wenn keine weiteren lokalen Marken mit dem selben Metrikbetrag vorhanden sind. Die Hilfsprozedur in Abbildung 3.22 beschreibt dies. Dargestellt ist zudem die Überprüfung von *GM*. *GM* kann sich indessen lediglich nach einer Markensuche ändern und wird nur deshalb überprüft, da die Hilfsprozedur *CheckGroupInformation* möglichst allgemein verfasst wurde, um auch in später beschriebenen Situationen verwendet werden zu können. Wird der Metrikbetrag der lokalen Marke erhöht, muss die Änderung an *LM* auch an den Vaterknoten im Kontrollbaum weitergegeben werden. Dazu wird die Nachricht *Update* gesendet, die in Abbildung 3.20 beschrieben wurde.

In Abbildung 3.21b ist diese Situation beispielhaft dargestellt. Klient $N_{12}$ sendet eine *repJoinGroup*

```
R empfängt repJoinGroup(G, N, B) von Klient C:
(1)   Reg := Gruppenregister mit Reg.GR == G;
(2)   if (Reg does not exist) or (Reg.LM > Reg.GM + Toleranz) then
(3)       sende TokenSearch(G, N, B, Reg.LM, R) an Vaterknoten;
(4)   else
(5)       TP := Markenpaket für Gruppe G mit TP.M == Reg.LM;
(6)       TP.Anzahl := TP.Anzahl - 1;
(7)       sende Token an N;
(8)       erstelle neues Markenpaket für Gruppe G und Eigentümer N mit Anzahl B Marken;
(9)       CheckGroupInformation;

Hilfsprozedur CheckGroupInformation: //Aktualisieren des Markencontainers und Gruppenregisters
(10)      if TP.M > Reg.GM + Toleranz then
(11)          Reg.GM := TP.M;
(12)      Reg.LM := min(∀ Markenpakete P der Gruppe G: P.M);
(13)      if Reg.GM or Reg.LM has changed then
(14)          send Update(G, Reg.LM, Reg.GM) an Vaterknoten;
(15)      if TP.Anzahl == 0 then
(16)          lösche TP;
(17)      if TC.SP == ∅ des Markencontainers TC mit TC.GR == G then
(18)          lösche TC;
```

*Abbildung 3.22: Beitritt zu einer Gruppe in TRS-M*

Aufforderung an seinen lokalen RepServer $S_1$. Die Belegung von *LM* und *GM* besagt, dass eine ideale Marke lokal vorhanden ist und aufgrund dessen eine Markensuche überflüssig ist. Abbildung 3.23a zeigt den Markencontainer nach dem Erzeugen der neuen Marken für den Eigentümer $N_{12}$. Eine Aktualisierung der Gruppenregister ist nicht notwendig, da $S_1$ eine weitere Marke mit Metrikbetrag eins besitzt.

Ist im Gegensatz dazu bei einem lokalen RepServer *R* die Situation gegeben, dass $LM > GM + Toleranz$ ist, dann ist keine ideale Marke lokal vorhanden. Mit einer Markensuche wird infolgedessen versucht, eine ideale Marke zu finden. Zu beachten ist, dass *GM* mit Ausnahme des Wurzel-RepServers lediglich eine untere Schranke des globalen Metrikbetrags darstellt (siehe Abschnitt 3.5.6.1). Deshalb ist es möglich, dass eine Markensuche erfolglos sein kann, d.h. keine Marke mit kleinerem Metrikbetrag als eine lokal vorhandene findet. Der hier beschriebene Algorithmus arbeitet korrekt mit der unteren Schranke des minimalen globalen Metrikbetrags, was schließlich ein leichtgewichtigeres Protokoll ermöglicht, da keine nebenläufigkeitsbedingten Konsistenzprobleme zu beachten sind. Dies resultiert zudem in einem größeren erzielbaren Durchsatz des TRS-Diensts. Während der Markensuche wird erkannt, wenn global keine Marke mit kleinerem Metrikbetrag als beim lokalen RepServer *R* gefunden werden kann und schließlich eine Marke von *R* geliefert.

In der ersten Suchphase, der Suchraumausdehnung, wird der überdeckte Teilbaum durch Weiterleiten der Suchnachricht an den Vaterknoten im TRS-Baum Schritt für Schritt ausgedehnt, bis ein RepServer gefunden wird, der mindestens eine der folgenden Bedingungen erfüllt: (1) $LM \leq GM + Toleranz$

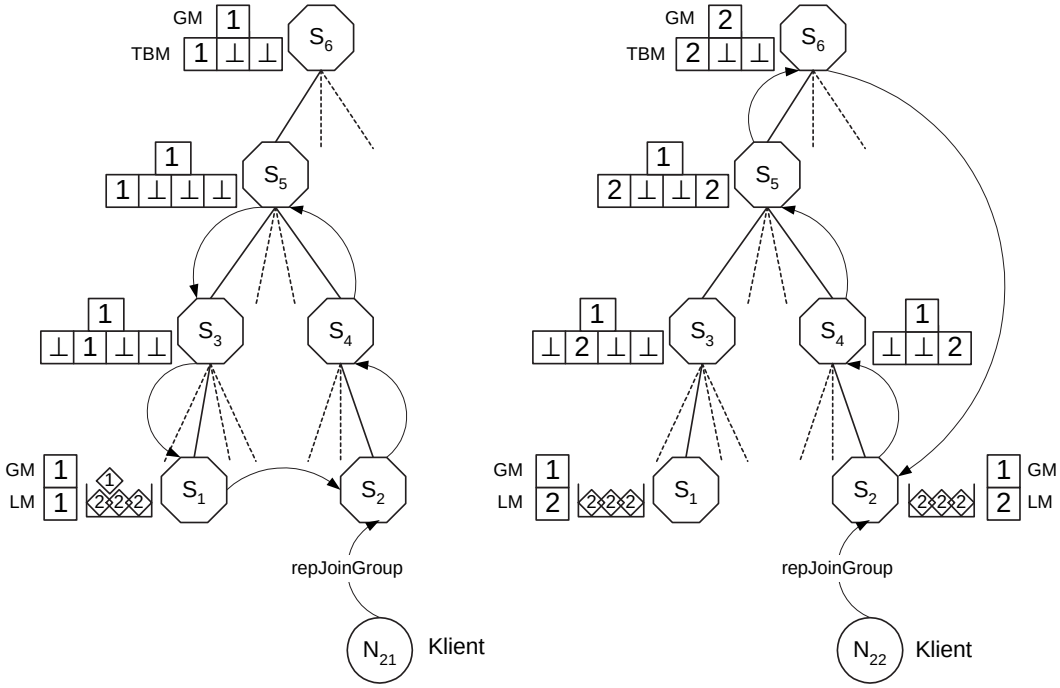



*Abbildung 3.23: (a) Beitritt zu einer Gruppe mit erfolgreicher Markensuche und (b) Beitritt zu einer Gruppe mit erfolgloser Markensuche in TRS-M*

des suchenden RepServers oder (2) $min(TBM) \leq GM + Toleranz$ (siehe Abbildung 3.24, Zeile 9 und Zeile 11). Tritt diese Situation ein, muss der Suchraum nicht weiter ausgedehnt werden, da sich eine ideale Marke bereits in dem überdeckten Teilbaum befindet. Ist die erste Bedingung erfüllt, so kann direkt auf eine lokale Marke des suchenden Blatt-RepServers *R* zurückgegriffen werden, da auch global keine geeignetere Marke gefunden werden kann. Ansonsten kann die zweite Suchphase, namentlich die Suchraumeinschränkung, erfolgen. Die Suchnachricht wird Schritt für Schritt an den Kind-RepServer *i* weitergeleitet, für den gilt: $TBM[i] = min(TBM)$, d.h. dessen Teilbaum eine ideale Marke mit kleinstem Metrikbetrag enthält (siehe Abbildung 3.24, Zeile 17). Die Übergabe der gefundenen Marke erfolgt wie in Strategie TRS-K direkt, d.h. ohne Umweg über die TRS-Hierarchie.

Anhand mehrerer Beispiele soll der Algorithmus nochmals verdeutlicht werden. Für alle Beispiele wird angenommen, dass die *Toleranz* bei der Markensuche null beträgt. In Abbildung 3.23a wird bereits beim Erreichen des ersten RepServers $S_5$, der zum Gruppenbaum gehört, die Suchrichtung gewechselt, da $min(TBM) = GM = 1$ ist. Der Markencontainer und die Gruppenregister nach der notwendigen Aktualisierung sind in Abbildung 3.23b dargestellt. Die *TBM*-Felder der Gruppenregister mussten auf dem Pfad von $S_1$ bis $S_6$ angepasst werden, da $S_1$ keine Marken mit Metrikbetrag eins mehr besitzt. Andererseits mussten zudem die *TBM*-Felder der Gruppenregister auf dem Pfad von $S_2$ bis $S_5$ angepasst werden, da $S_2$ nun ebenfalls Marken des Metrikbetrags zwei besitzt. *GM* aller Gruppenregister außer der Wurzel enthält derweil weiterhin den Metrikbetrag eins, da in einem anderen

```
R empfängt TokenSearch(G, N, B, LM, L) von Kindknoten S:
(1)  Reg := Gruppenregister für Gruppe G;
(2)  if R is Wurzel-RepServer then
(3)      if (Reg does exist) and (min(∀ i Reg.TBM[i]) < LM) then
(4)          sende TokenSearch(G, N, B, LM, L) an Kind i mit Reg.TBM[i] == min(∀ i Reg.TBM[i]);
(5)      else
(6)          sende TakeLocalToken(G, N, B, LM) an L;
(7)  else
(8)      case
(9)          | (Reg exists) and LM ≤ Reg.GM + Toleranz then
(10)             sende TakeLocalToken(G, N, B, LM) an L;
(11)         | (Reg exists) and (min(∀ i Reg.TBM[i]) ≤ Reg.GM + Toleranz) then
(12)             sende TokenSearch(G, N, B, LM, L) an i mit Reg.TBM[i] == min(∀ i Reg.TBM[i]);
(13)     else
(14)         sende TokenSearch(G, N, B, LM, L) an Vaterknoten;

R empfängt TokenSearch(G, N, B, LM, L) von Vaterknoten S:
(15) Reg := Gruppenregister für Gruppe G;
(16) if R != Blatt-RepServer then
(17)     sende TokenSearch(G, N, B, LM, L) an i mit Reg.TBM[i] == min(∀ i Reg.TBM[i]);
(18) else
(19)     TP := Markenpaket für Gruppe G mit TP.M == Reg.LM;
(20)     TP.Anzahl := TP.Anzahl - 1;
(21)     sende Token(G, N, B, TP.M) an L;
(22)     CheckGroupInformation; // siehe Abbildung 3.22

R empfängt Token(G, N, B, M) von RepServer S:
(23) sende Token an N;
(24) erstelle Markenpaket und gegebenenfalls Markencontainer
         für Gruppe G mit Eigentümer N und Anzahl B Marken;
(25) CheckGroupInformation; // siehe Abbildung 3.22
```

*Abbildung 3.24: Markensuche in TRS-M*

Teilbaum noch Marken des Metrikbetrags eins existieren können. Dies kann einzig durch eine globale Markensuche bis zum Wurzel-RepServer entschieden werden, die bisher noch nicht durchgeführt wurde. Diese globale Markensuche ist in Abbildung 3.23b dargestellt. Alle passierten RepServer $S_2$, $S_4$ und $S_5$ zeigen das mögliche Vorhandensein einer Marke mit Metrikbetrag eins an, können in ihren Teilbäumen dagegen nur Marken des Metrikbetrags zwei anbieten. Erst beim Erreichen der Wurzel kann entschieden werden, dass Marken des Metrikbetrags eins tatsächlich nirgends mehr vorhanden sind. Da die lokalen Marken von $S_2$ den gleichen Metrikbetrag aufweisen wie die bei $S_1$, wird direkt eine Marke von $S_2$ ausgewählt. $S_2$ veranlasst nun die Aktualisierung der globalen Metrikbeträge *GM* auf dem Pfad von $S_2$ bis $S_5$. Eine globale Anpassung aller Gruppenregister erfolgt nicht, da sonst auch inaktive Bereiche des Gruppenbaums einen unnötigen Anpassungsaufwand hätten. Die aktualisierten Gruppenregister sind in Abbildung 3.25a dargestellt. Eine weitere Suche ausgehend von $S_1$ stößt beim Erreichen von $S_5$ auf den aktualisierten globalen Metrikbetrag und erkennt, dass eine wei-

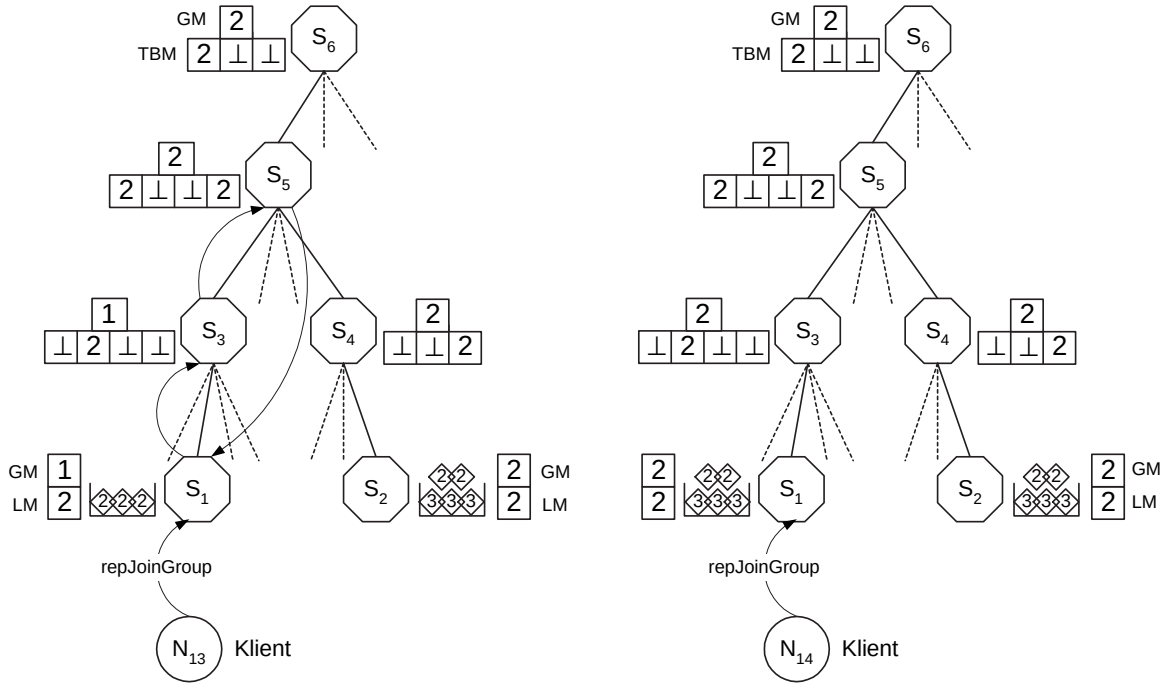



*Abbildung 3.25: (a) Beitritt zu einer Gruppe mit erfolgloser Markensuche und (b) Gruppenregister nach dem Aktualisieren in TRS-M*

tere Suchraumausdehnung nicht erforderlich ist. Abbildung 3.25b zeigt die Gruppenregister nach der Aktualisierung von $S_1$.

#### 3.5.6.4 Gruppenaustritt

Der Gruppenaustritt geht analog zur Strategie TRS-K vonstatten, d.h. es werden alle Marken des austretenden Teilnehmers gelöscht und die selbst belegte Marke des Vaterknotens im Kontrollbaum wieder dem Repository zugeführt. Dagegen ergibt sich eine wesentliche Komplikation nachfolgender Markensuchen dadurch, dass nun wieder eine Marke im Repository vorhanden sein kann, dessen Metrikbetrag niedriger ist als der globale Metrikbetrag *GM* der Gruppenregister anzeigt. Um diesen Zustand zu korrigieren genügt es nicht, nur *GM* und *TBM* auf dem Pfad des betroffenen Blatt-RepServers bis zum Wurzel-RepServer mit einer *Update*-Nachricht auf den Metrikbetrag der wiedererhaltenen Marke zu setzen. Dadurch kann zwar eine Markensuche, die den Wurzel-RepServer erreicht, die ideale Marke finden. Manche Markensuchen würden derweil bereits früher, vor dem Erreichen des Wurzel-RepServers die Suchraumausdehnung beenden, da der globale Metrikbetrag *GM* keinen Erfolg verspricht. Folglich muss *GM* aller Gruppenregister aktualisiert werden. Die *Update*-Operation, wie in Abbildung 3.20 dargelegt, wird dazu erweitert und überprüft nun, ob sich *GM* durch die erhaltene Nachricht erniedrigt. Ist dies der Fall, wird *Update* nicht nur in Richtung der Wurzel,

| Nachricht | Typ | Beschreibung |
|---|---|---|
| Update(G, TBM, GM) | Unicast | Einrichten eines Gruppenregistereintrags für Gruppe *G*. *TBM* ist der minimale Metrikbetrag des sendenden Teilbaums und *GM* ist dessen globaler Metrikbetrag. |
| DeleteGroup(G) | Unicast | Löschen der Gruppe *G*. |
| TokenSearch(G, N, B, LM, L) | Unicast | Suchen einer Marke der Gruppe *G* ausgehend von RepServer *L*. Der neue Teilnehmer *N* hat einen maximalen Verzweigungsgrad von *B*. Der kleinste lokale Metrikbetrag aller Marken der Gruppe *G* bei *L* beträgt *LM*. |
| TakeLocalToken(G, N, B, LM) | Unicast | Aufforderung eine lokale Marke zu wählen. Bedeutung der Parameter siehe oben. |
| Token(G, N, B, M) | Unicast | Übergabe einer Marke mit Metrikbetrag *M*. *N* und *B* beziehen sich auf den neuen Teilnehmer (siehe oben). |

*Tabelle 3.3: Interne Nachrichten von TRS-M*

sondern auch in Richtung aller Blatt-RepServer weiterpropagiert.

In Abbildung 3.3 finden sich alle internen Operationen von TRS-K nochmals in der Zusammenfassung.

## 3.6 Protokollverhalten in Fehlersituationen

In diesem Abschnitt wird das Verhalten des Token-Repository-Service in möglichen Fehlersituationen dargelegt. Das Fehlermodell aus Abschnitt 3.5.3 wird zu diesem Zweck erweitert und schließt nun Knoten– und Kommunikationsfehler ein. Nach der Definition des Fehlermodells im folgenden Abschnitt wird das Protokollverhalten in diesen Fehlersituationen aufgezeigt und die Konsequenzen für die Fehlersemantik erläutert.

### 3.6.1 Fehlermodell

Das Systemmodell aus Abschnitt 3.5.3 soll unverändert übernommen werden. Zusammengefasst wird ein asynchrones Kommunikationssystem angenommen, das autonome Rechnerknoten verbindet. Das Fehlermodell unterscheidet Knoten– und Kommunikationsfehler.

Die möglichen Arten von Knotenfehlern lassen sich in Fehlerklassen unterteilen [Cristian 1991]. Der folgenden Beschreibung liegt die Annahme der Fehlerklasse *Zusammenbruchsfehler* (engl. Crash) zugrunde. Bei Zusammenbruchsfehlern arbeitet ein Knoten entweder korrekt oder er hält vollständig

an und bleibt in diesem Zustand. Der Wiederanlauf des Knotens kann somit kontrolliert erfolgen. Zur Fehlerklasse der Zusammenbruchsfehler gehört auch die Annahme, dass andere Knoten nicht in der Lage sind, einen Knotenausfall zweifelsfrei festzustellen. In einem asynchronen System ist dies eine notwendige Annahme, da ausgefallene Knoten nicht von langsamen Knoten oder einem langsamen Kommunikationssystem unterschieden werden können.

Ein Knoten kann flüchtigen und stabilen Speicher besitzen. Der Inhalt von flüchtigem Speicher kann bei einem Knotenausfall verloren gehen. Der Inhalt von stabilem Speicher bleibt auch bei Knotenausfall erhalten.

Für Kommunikationsfehler wird die Annahme getroffen, dass die üblichen Mechanismen bestehend aus Quittungen, Zeitgebern und Übertragungswiederholungen für eine zuverlässige Kommunikation sorgen. Die Kommunikationspartner müssen sich dazu allerdings in der selben Netzpartition befinden. Existiert mehr als eine Netzpartition und befinden sich beide Kommunikationspartner in unterschiedlichen Netzpartitionen ist keine zuverlässige Kommunikation möglich. Unter der Annahme, dass Fehlersituationen nicht dauerhaft sind, ist in endlicher Zeit die Kommunikation wieder möglich.

Im nächsten Abschnitt wird das Verhalten des TRS-Diensts unter der Annahme des dargelegten Fehlermodells beschrieben.

### 3.6.2 Protokollerweiterungen zur Behandlung von Fehlern

Die folgenden Beschreibungen basieren auf dem TRS-M-Dienst. Für die einfacheren Verfahren TRS-K und TRS-R können analoge Protokollerweiterungen zur Behandlung von Fehlern vorgenommen werden.

RepServer nutzen flüchtigen Speicher, dessen Inhalt bei Knotenausfall verloren gehen kann, um alle häufig veränderlichen Informationen zu speichern. Dazu gehören die Marken und die Gruppenregister. Deren Speicherung auf stabilen Speicher würde den Durchsatz des TRS-Diensts verringern. Die einzigen Daten auf stabilem Speicher betreffen die TRS-Hierarchie, d.h. die Kenntnis des Vaterknotens und aller Kindknoten.

Erfährt ein RepServer einen Zusammenbruch, so wird nach endlicher Zeit ein Wiederanlauf veranlasst. Dabei werden die Inhalte des stabilen Speichers wiederhergestellt, d.h. die Kenntnis über Vater– und Kindknoten in der TRS-Hierarchie, während der Inhalt des flüchtigen Speichers vorerst verloren bleibt. Dennoch werden nach endlicher Zeit die Marken der Blatt-RepServer und die Gruppenregister der Such-RepServer wie folgt wiederhergestellt. Wird eine periodisch gesendete *repRefreshToken*-Nachricht eines Markeneigentümers empfangen, so verlängert der RepServer nicht nur die Ablaufzeit des betreffenden Markenpakets entsprechend. Für den Fall, dass das Markenpaket nicht (mehr) existiert, legt er es neu an. Verlorene Gruppenregister werden in ähnlicher Weise wiederhergestellt.

Dies erfolgt aufgrund einer empfangenen *Update*-Nachricht eines Kindknotens. Erfährt ein Knoten längere Zeit keine Änderung an seinen lokalen Marken oder seinen Gruppenregistern und sendet infolgedessen auch keine reguläre *Update*-Nachricht an seinen Vaterknoten, so setzt vor dem Erreichen der Ablaufzeit ein periodischer Mechanismus ein, um eine Mindestfrequenz an *Update*-Nachrichten zu gewährleisten.
Aus der Sichtweise der RepServer ergibt sich folgendes Verhalten. Wird ein Blatt-RepServer, vor dem vollständigen Wiederherstellen aller Marken, um eine nicht vorhandene Marke gefragt, so startet er eine Markensuche. Dies ist das reguläre Verhalten eines Blatt-RepServers, der auch prinzipiell keine Kenntnis davon hat, ob alle Marken bereits wiederhergestellt wurden. Ist dieser Blatt-RepServer andererseits das Ziel einer Markensuche innerhalb der RepServer-Hierarchie, dann wird dem Vaterknoten mitgeteilt, dass keine Marken zur Verfügung stehen und die Markensuche erneut gestartet. Ein noch nicht vollständig wiederhergestellter Such-RepServer leitet Markensuchen der Kindknoten an seinen Vaterknoten weiter. Wurde die Suchnachricht bereits von seinem Vaterknoten gesendet und kann diese aufgrund fehlender Gruppenregister nicht an einen Kindknoten weitergeleitet werden, so wird dem Vaterknoten eine *Update*-Nachricht zur Änderung seines Gruppenregisters gesendet. Darauf folgend wird erneut die erste Markensuchphase in Richtung Vaterknoten eingeleitet.
Die Sichtweise eines Klienten ist wie folgt. Verläuft eine Markensuche aufgrund eines ausgefallenen RepServers oder einer Netzpartitionierung innerhalb der TRS-Hierarchie erfolglos, so wählt der Klient einen anderen RepServer, vorzugsweise aus einer nahe gelegenen Domäne und wiederholt die Markenanfrage. Nicht nur für die Operation *repJoinGroup*, sondern auch für *repCreateGroup* und *repDeleteGroup* wird diese Vorgehensweise verfolgt. Dagegen ist dies bei *repLeaveGroup* nicht möglich. Ist der lokale RepServer nicht erreichbar, wird *repLeaveGroup* einfach ausgelassen, da das Löschen von Marken einzig durch den RepServer durchführbar ist, der die Marken auch angelegt hat. Die Verwendung ungültiger Marken wird indes spätestens vom neuen Teilnehmer bemerkt, der sich nicht erfolgreich an den referenzierten Eigentümer anmelden kann. Der neue Teilnehmer fragt daraufhin nach einer neuen Marke, mittels einer weiteren *repJoinGroup*-Nachricht. Nach einiger Zeit befindet sich die gesamte Markeninformation wieder in einem konsistenten Zustand, da die Marken entweder durch den Serverausfall gelöscht wurden, ein neuer Teilnehmer die inkonsistenten Marken verbraucht hat oder aber die Ablaufzeit der Marken erreicht wurde.
Die Gruppenerstellung und der Gruppenbeitritt an einen anderen RepServer hat zur Folge, dass auch ein gegebenenfalls später durchzuführender Gruppenaustritt und alle *repRefreshToken*-Operationen dort zu erfolgen haben. Alternativ kann die Gruppenerstellung oder Gruppenlöschung auch komplett unterbleiben. Für die Gruppenerstellung ist dies äquivalent zu dem Fall, dass alle Marken verloren gehen (siehe unten). Unterbleibt die Gruppenlöschung, werden die ungültigen Marken durch neue Teilnehmer oder schließlich mit Hilfe der Ablaufzeit gelöscht.
In seltenen Fällen ist es denkbar, dass kein RepServer von einem neuen Teilnehmer aus erreichbar ist bzw. dass momentan keine Marken im TRS-Dienst verfügbar sind. In dieser Situation kann ein

neuer Teilnehmer auf reguläres ERS zurückgreifen, um nach einem Gruppenführer im Kontrollbaum zu suchen. Nach dem erfolgreichen Anmelden übergibt der neue Teilnehmer, sofern dies möglich ist, Marken an seinen lokalen RepServer mittels *repAddToken*. Damit können nachfolgende *repJoinGroup*-Operationen wieder über den effizienteren TRS-Dienst erfolgen und eine nahtlose Integration von ERS in den TRS-Dienst wird erzielt.

Zusammenfassend lassen sich zwei mögliche Auswirkungen beim Auftreten von Fehlern identifizieren: RepServer können nicht verfügbar und Marken– oder Gruppeninformationen können veraltet sein oder fehlen. Durch das Ausweichen auf alternative RepServer, durch periodische Aktualisierungsnachrichten und durch das Erkennen veralteter Gruppen– und Markeninformation durch den Ablaufzeitmechanismus oder spätestens beim Anmelden an den Eigentümer wird darauf reagiert. Ist trotz der aufgezeigten Maßnahmen kurzzeitig keine Marke über den TRS-Dienst zu erhalten, so wird auf ERS ausgewichen.

Aus den beschrieben Maßnahmen ergibt sich schließlich folgende Konsequenz für die Fehlersemantik. Innerhalb des gewählten Fehlermodells kann sich ein neuer Teilnehmer an eine Multicast-Gruppe anmelden, wenn sich der neue Teilnehmer und mindestens ein unvollständig verzweigter Knoten der gesuchten Multicast-Gruppe in der selben Netzpartition befinden. Diese Fehlersemantik ist äquivalent zur Fehlersemantik von ERS und ERA.

## 3.7 Alternativen für die Verwendung der Metrik

Die in den vorhergehenden Abschnitten verwendete beispielhafte Metrik, die Höhe des Markeneigentümers im Kontrollbaum, hat es ermöglicht, Kontrollbäume mit minimaler Höhe zu erzeugen. In diesem Abschnitt werden alternative Metriken beschrieben, die ebenfalls große Bedeutung beim Kontrollbaumaufbau haben.

### 3.7.1 Optimierung nach Verzögerung

Eine alternative Metrik für den TRS-Dienst ist die Optimierung nach Verzögerung im Kontrollbaum. Die Auslieferung der initialen Multicast-Nachricht unterliegt nicht dem Einflussbereich des TRS-Diensts, sondern nur der physischen und logischen Lage eines Knotens im Netzwerk und den Routing-Protokollen. Daher kann die initiale Verzögerung auch nicht beeinflusst werden. Dagegen wird die Verzögerung für potentielle Übertragungswiederholungen durch den Kontrollbaum mitbestimmt. Wie schnell eine mögliche Übertragungswiederholung gesendet werden kann, hängt von vielen Parametern ab, z.B. dem Multicast-Transportprotokoll, den Einstellungen der Zeitgeber, den Abständen in einer lokalen Gruppe usw. und schließlich auch von der Nachrichtenverzögerung zum Vaterknoten im

Kontrollbaum und dessen eigene Verzögerung für eine Nachrichtenauslieferung. Der letzte Einflussparameter, nämlich die Verzögerung für die Nachrichtenauslieferung des Vaterknotens, kann durch die Wahl einer geeigneten Metrik berücksichtigt werden. Vorausgesetzt wird hier, dass die Verzögerungszeiten im Netzwerk hinreichend konstant bleiben, da ansonsten eine Optimierung der Verzögerung nur zu einem bestimmten Zeitpunkt nicht sinnvoll ist. Die Verzögerung des neuen Teilnehmers zum Vaterknoten könnte einzig durch Messung der Verzögerungen zu mehreren oder allen potentiellen Vaterknoten ermittelt werden, was einen beträchtlichen Aufwand darstellen würde und deshalb nicht berücksichtigt werden soll.

Der Wurzelknoten besitzt keine Verzögerung der Nachrichtenauslieferung, demnach wird diesem Knoten der Metrikbetrag null zugeordnet. Die direkten Kindknoten der Wurzel messen ihre Nachrichtenverzögerung zur Wurzel, z.B. durch eine Ping-Messung (siehe [Postel 1981a], dort allerdings als Echo anstatt Ping bezeichnet) und setzen ihren Metrikbetrag auf die der Nachrichtenverzögerung. Prinzipiell misst jeder Knoten die Verzögerung zu seinem Vaterknoten und addiert den Metrikbetrag des Vaterknotens mit der Verzögerung, um den eigenen Metrikbetrag zu erhalten. Eine Auswahl der Marken mit Rücksicht auf diese Metrik liefert vorwiegend oder garantiert, je nach verwendeter TRS-Strategie, den Vaterknoten mit geringster kumulierter Verzögerung zum Wurzelknoten, was bei Nachrichtenverlust mehrerer Gruppenführer auf dem Pfad zwischen einem Empfänger und der Wurzel maßgeblich die Zeit für Übertragungswiederholungen beeinflusst (siehe Abschnitt 4.5.4). Weiterhin werden bei gleichem Metrikbetrag Vaterknoten der eigenen Domäne oder mit geringer Distanz im Netzwerk bevorzugt. Die entstehenden Kontrollbäume erlauben somit geringe Verzögerungen für Übertragungswiederholungen.

### 3.7.2 Optimierung für einen Sender

Das bisher betrachtete überaus allgemeine Modell einer zuverlässigen Multicast-Gruppe aus Abschnitt 3.3 erlaubt beliebig viele Sender pro Gruppe und ist eine wichtige Voraussetzung dafür, dass der TRS-Dienst in beliebige Transportprotokolle integriert werden kann. Andererseits sind viele Anwendungen nur auf einen Sender angewiesen, dazu gehört beispielsweise die Informationsverteilung, oder aber das Routing-Protokoll erlaubt gar nur einen Sender (siehe PIM-SSM [Bhattacharyya et al. 2000a, Bhattacharyya et al. 2000b] in Abschnitt 2.4.1.5). Dieser Sonderfall ist selbstredend durch das allgemeine Modell abgedeckt. Dennoch lohnt es sich, gerade weil es einen häufigen Sonderfall darstellt, dem weitere Aufmerksamkeit zu schenken. Eine sinnvolle Optimierung kann durch geschickte Verwendung der Metrik vom TRS-Dienst erbracht werden. Das Ziel ist es zu erreichen, dass Gruppenführer gewählt werden, die bezüglich der Netzwerkdistanz näher zum Sender liegen als deren Kindknoten. Damit besitzen Gruppenführer eine höhere Wahrscheinlichkeit, eine Nachricht zuverlässig zu empfangen, was schnellere und effizientere Übertragungswiederholungen erlaubt (siehe auch Diskussion in Abschnitt 3.2.3).

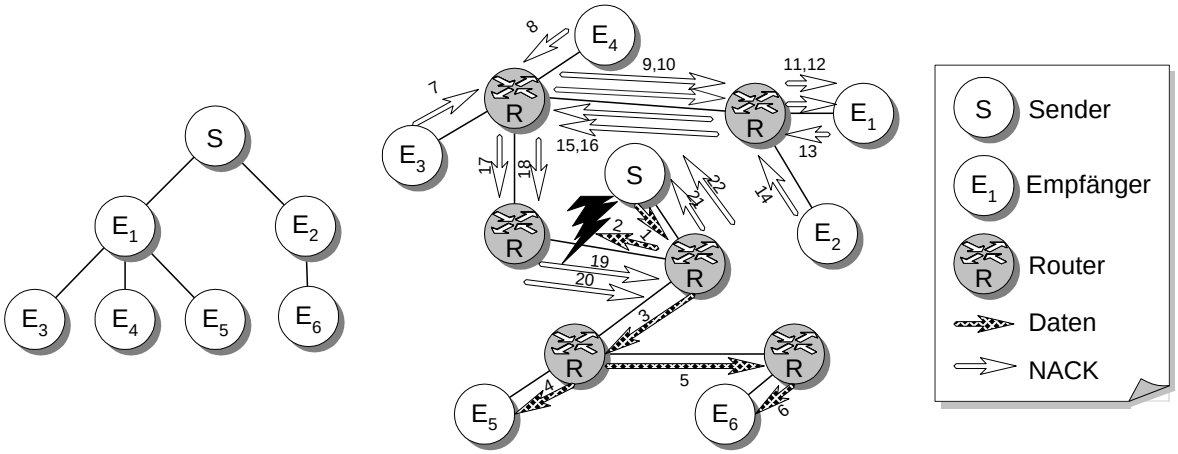



*Abbildung 3.26: Kontrollbaum (links) und Nachrichten im Netzwerk bei Datenverlust (rechts)*

Abbildung 3.26 zeigt beispielhaft einen Kontrollbaum und die zugehörige Struktur des Netzwerks. Der Sender sendet eine Nachricht an die Multicast-Gruppe, wobei ein Datenverlust bei der Nachricht zwei auftritt. Der ineffiziente Kontrollbaum zeigt sich durch die anschließenden negativen Quittungen, die von den Empfängern $E_3$ und $E_4$ in Richtung ihres Vaterknotens $E_1$ gesendet werden. Dies ist ineffizient, da der Vaterknoten $E_1$ über einen gemeinsamen Pfad mit $E_3$ und $E_4$ an den Sender angebunden ist, der Nachrichtenverlust auf diesem gemeinsamen Pfad auftrat, aber $E_1$ weiter vom Sender entfernt ist als seine Kindknoten. Deshalb werden die negativen Quittungen zuerst in die falsche Richtung, noch weiter weg vom Sender gesendet, obwohl $E_1$ die Daten nicht liefern kann und seinerseits eine Nachrichtenwiederholung vom Sender anfordert. In Abbildung 3.27 ist eine alternative Organisation des Kontrollbaums illustriert, welche die gleiche Situation mit lediglich 12 anstatt 16 negativen Quittungen behandelt. Auch das hier nicht dargestellte Senden der Übertragungswiederholungen kann dadurch effizienter erfolgen. Diese Verbesserung wurde erreicht, indem Vaterknoten gewählt wurden, die eine kleinere Distanz zum Sender aufweisen als die Kindknoten.

Diese Erkenntnis kann für den Aufbau des Kontrollbaums eingesetzt werden. Ohne Router-Unterstützung müssen dabei Abstriche gemacht werden, was die zur Verfügung stehenden Informationen über die Netztopologie betrifft. Erfreulicherweise reicht bereits die Kenntnis über den Abstand zur Wurzel, ausgedrückt in der Anzahl Teilstrecken, um eine in vielen Fällen ausreichende Optimierung des Kontrollbaums durchzuführen. Die Teilstreckenanzahl kann einfach ermittelt werden, z.B. mittels Traceroute [Malkin 1993, Stevens 1994] oder auch Ping [Postel 1981a], welches mit beschränktem aber zunehmendem TTL gesendet wird, bis eine Antwort eintrifft.

Die Teilstreckenanzahl eines Markeneigentümers zum Sender kann direkt als Metrikbetrag verwendet werden. Der TRS-Dienst soll eine Marke mit einer kleineren Teilstreckenanzahl, als das neue Mitglied besitzt, liefern. Mittels TRS-R und TRS-K, die lediglich eine lokale Optimierung bezüglich

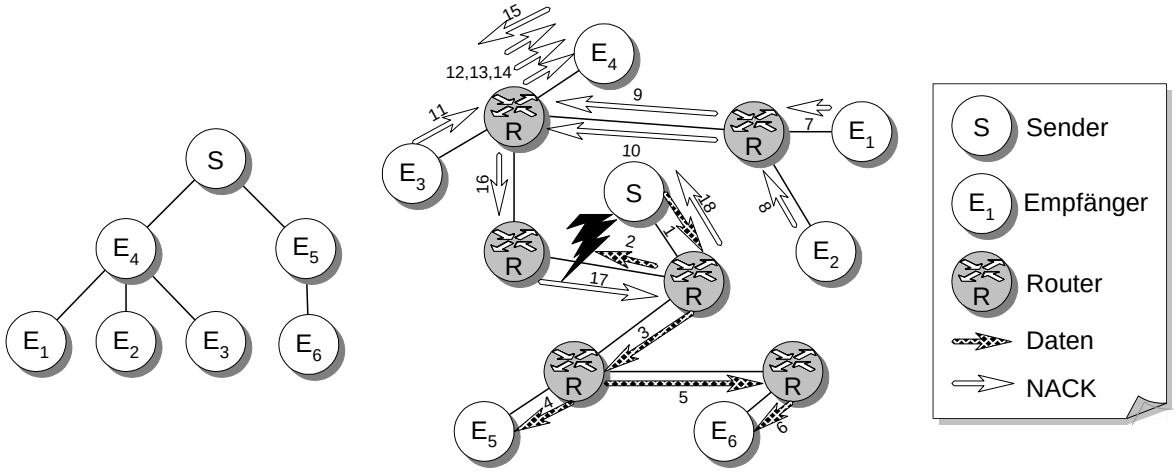



*Abbildung 3.27: Optimierter Kontrollbaum (links) und Nachrichten im Netzwerk bei Datenverlust (rechts)*

der Metrik unterstützen, ist dies nur beschränkt möglich, während mittels TRS-M eine globale Suche durchgeführt werden kann. Damit kann eine sinnvolle Optimierung des Kontrollbaums für einen Sender durchgeführt werden.

## 3.8 Speicherverbrauch und Durchsatz des Token-Repository-Service

Ausgehend von einer abstrakten Architektur und Schnittstelle wurden in den vorigen Abschnitten verschiedene Strategien für eine Umsetzung des Token-Repository-Service aufgezeigt. Für die Beurteilung, ob auch ein praktischer Einsatz erfolgen kann, erscheint es zusätzlich notwendig, den Speicherverbrauch und den Durchsatz zu diskutieren.

### 3.8.1 Speicherverbrauch

Bei der Beschreibung des Token-Repository-Service wurde bisher der Speicherverbrauchs noch nicht berücksichtigt. Die folgende einfache Berechnung soll helfen, den benötigten Speicher abzuschätzen.

Sei $S_C$ der Speicherverbrauch für einen Markencontainer und sei $S_P$ der Speicherverbrauch für ein Markenpaket. Die Anzahl der Multicast-Gruppen sei $G$ und die Anzahl der Teilnehmer pro Multicast-Gruppe sei $R$. Der Gesamtspeicherverbrauch $S$ für alle Marken ist folgendermaßen berechenbar:

$$S = GS_C + GRS_P. \tag{3.1}$$

| | | |
|---|---|---|
| $G$ | - | Anzahl der Multicast-Gruppen. |
| $R$ | - | Anzahl der Teilnehmer pro Multicast-Gruppe. |
| $L$ | - | Anzahl an Blatt-RepServern. |
| $M_{neu}$ | - | Erstellungsrate neuer Multicast-Gruppen. |
| $A$ | - | Aktualisierungsrate der Gruppenregister. |
| $B$ | - | Verzweigungsgrad des Kontrollbaums. |
| $h$ | - | Höhe des erstellten Kontrollbaums. |
| $D$ | - | Last des Wurzel-RepServers. |
| $S_C$ | - | Speicherverbrauch für einen Markencontainer. |
| $S_P$ | - | Speicherverbrauch für ein Markenpaket. |
| $S_R$ | - | Speicherverbrauch für ein Gruppenregister. |
| $S$ | - | Gesamtspeicherverbrauch für alle Marken im TRS-Dienst. |
| $S_{Blatt}$ | - | Mittlerer Speicherverbrauch pro Blatt-RepServer. |
| $S_{Such}$ | - | Mittlerer Speicherverbrauch pro Such-RepServer. |

*Tabelle 3.4: Notation für die Analyse von Speicherverbrauch und Durchsatz des TRS-Diensts*

Die Notation ist in Tabelle 3.4 zusammengefasst. Generell bestehen Variablen immer aus einem Buchstaben, so dass keine Verwechslung zwischen einer Multiplikation und längeren Variablenbezeichnern möglich ist.

Die Anzahl bereits verbrauchter und folglich gelöschter Markenpakete kann nicht ermittelt werden, da im ungünstigsten Fall alle Markenpakete noch mindestens eine Marke beinhalten und somit keines gelöscht werden kann. Formel 3.1 stellt deswegen eine obere Schranke des Speicherverbrauchs dar. Mit Hilfe von $L$, das die Anzahl der Blatt-RepServer angibt, kann der mittlere Speicherverbrauch pro Blatt-RepServer $S_{Blatt}$ ermittelt werden:

$$S_{Blatt} = \frac{S}{L}. \tag{3.2}$$

Mit der Größe eines Gruppenregisters $S_R$ ist der Speicherverbrauch eines Such-RepServers $S_{Such}$ maximal:

$$S_{Such} = GS_R. \tag{3.3}$$

Formel 3.3 liegt die Annahme zugrunde, dass jeder Such-RepServer für alle Multicast-Gruppen ein Gruppenregister gespeichert hat, was wiederum eine obere Schranke des Speicherverbrauchs darstellt.

Im Folgenden wird für ein Szenario beispielhaft der Speicherverbrauch dargelegt. Es wird angenommen es existieren eine Million Multicast-Gruppen (=$G$) mit jeweils 1000 Teilnehmern (=$R$). Ein Markencontainer verbraucht vier Byte Speicherplatz (=$T_C$), da die IP-Gruppenaddresse 32 Bit beträgt. Für

den Speicherverbrauch eines Markenpakets sollen acht Byte angesetzt werden (=$S_P$) bestehend aus vier Byte IP-Adresse des Eigentümers, ein Byte für den Metrikbetrag, ein Byte für die Anzahl der Marken und zwei Byte für die Ablaufzeit. Der Speicherverbrauch eines Blatt-RepServers bei angenommenen 1000 (=$L$) RepServern beträgt:

$$S_{Blatt} = \frac{1000000 \cdot 4 + 1000000 \cdot 1000 \cdot 8}{1000} \, Byte = 8 \, MByte. \tag{3.4}$$

Unter der Annahme, dass ein Gruppenregister für TRS-K fünf Byte Speicher belegt, bestehend aus vier Byte für die Gruppenadresse und einem Byte für die Teilbäume, beträgt der Speicherverbrauch eines Such-RepServers:

$$S_{Such} = 1000000 \cdot 5 \, Byte = 5 \, MByte. \tag{3.5}$$

Für den TRS-M-Dienst wird ein Byte für alle Teilbäume nicht ausreichen, da für jeden Teilbaum der Metrikbetrag abgespeichert werden muss. Unter der Annahme, dass die Speicherung des Metrikbetrags pro Teilbaum ein Byte belegt, wie es beispielsweise für die Speicherung der Markenhöhe ausreichend ist, und der Verzweigungsgrad zehn betragen soll, wird für ein Gruppenregister 14 Byte Speicher belegt. Das oben aufgeführte Szenario führt folglich zu 14 MByte Speicherverbrauch.

Die ermittelten Größen sind bereits von heutigen, handelsüblichen Rechnern problemlos zu bewältigen, während die Anzahl der Gruppen und der Teilnehmer bei weitem noch nicht erreicht ist. Für den Fall, dass sich Gruppenkommunikation explosionsartig verbreitet, soll dennoch ein Lösungsansatz für den gegebenenfalls zu hohen Speicherverbrauch gegeben werden. Dies wird im folgenden Abschnitt erfolgen, da damit ebenfalls eine Durchsatzsteigerung erfolgt.

### 3.8.2 Durchsatz

Die hierarchische Strukturierung der TRS-RepServer bei TRS-K und TRS-M ermöglicht eine äußerst effiziente Suche nach Marken. Für den praktischen Einsatz muss allerdings zudem die Frage gestellt werden, ob nicht gerade dadurch ein Flaschenhals entsteht, der besonders an der Baumwurzel ausgeprägt sein könnte (vgl. hierarchische Verbindungsnetzwerke von Parallelrechnern [Bräunl 1993]). Die folgende analytische Betrachtung ist eine Grundlage für die Beurteilung. Anschließend wird skizziert, wie eine Flaschenhalsbildung vermieden und eine Durchsatzsteigerung erzielt werden kann.

Blatt-RepServer können bei der Betrachtung außen vor gelassen werden. Sollten diese einen Flaschenhals bilden, so kann durch den Einsatz weiterer Blatt-RepServer und der damit einhergehenden Lastverteilung dieser Flaschenhals behoben werden. Die Last an Such-RepServern ergibt sich durch das Anlegen von Gruppen, die Suche nach Marken und die Aktualisierung der Gruppenregister. Beispielhaft wird die Situation des Wurzel-RepServers betrachtet. Sei $M_{neu}$ die Erstellungsrate neuer

Multicast-Gruppen, $G$ die Gesamtanzahl Multicast-Gruppen und $A$ die Aktualisierungsrate der Gruppenregister. Die Last des Wurzel-RepServers beträgt demnach pro Zeiteinheit für TRS-K:

$$D = M_{neu} + (B-1)M_{neu} + AG = BM_{neu} + AG. \tag{3.6}$$

Für jede neue Multicast-Gruppe muss eine Nachricht empfangen werden und in dessen Folge ein Gruppenregister angelegt werden. Für die Suche nach Marken muss im schlechtesten Fall für TRS-K von allen $B$ Kindknoten mit Ausnahme des Kindknotens, aus dessen Teilhierarchie die Gruppenerstellung initiiert wurde, eine Markensuche verarbeitet werden. Schließlich muss noch die Aktualisierung der Gruppenregister berücksichtigt werden. Das Löschen von Multicast-Gruppen soll nicht explizit betrachtet werden, da der Aufwand prinzipiell bereits in der Aktualisierungsrate der Gruppenregister beinhaltet ist.

Für TRS-M ist eine allgemeine Beurteilung der RepServer-Last diffizieler, da von der Häufigkeit globaler Markensuchen, die ihrerseits von der verwendeten Metrik abhängen, das Ergebnis beträchtlich beeinflusst wird. Deswegen soll die Beurteilung anhand des durchgängig verwendeten Beispiels, nämlich der Höhe des Markeneigentümers erfolgen. Eine weitere Annahme betrifft die Gleichmäßigkeit der Anmeldungen. Hier wird vorausgesetzt, dass diese gleichverteilt, sowohl zeitlich als auch räumlich, über allen RepServern erfolgen. Für jede Multicast-Gruppe sollen $R$ Anmeldungen erfolgen. Die Höhe jedes Kontrollbaums beträgt folglich (siehe Formel 3.19):

$$h = \log_B(R(B-1)+1), \tag{3.7}$$

was in $h-1$ Markensuchen und Aktualisierungen der Gruppenregister pro Teilbaum und Gruppe mündet, d.h. insgesamt $2G(h-1)B$ Anfragen. Die Last des Wurzel-RepServers für TRS-M beträgt damit pro Zeiteinheit:

$$D = M_{neu} + 2G(\log_B(R(B-1)+1)-1)B + AG. \tag{3.8}$$

Das folgende Szenario soll das Ergebnis praktisch veranschaulichen. Es wird angenommen, es existieren eine Million Multicast-Gruppen (=$G$). Pro Tag sollen 10000 Multicast-Gruppen neu erstellt werden (=$M_{neu}$) und pro Multicast-Gruppe 100 Anmeldungen erfolgen (=$R$). Die Aktualisierungsrate soll eine Stunde betragen (=$A$). Der Verzweigungsgrad soll zehn betragen (=$B$). Die Last des Wurzel-RepServers für TRS-K beträgt damit pro Sekunde:

$$D = \left(\frac{10000}{86400} + \frac{9 \cdot 10000}{86400} + \frac{1000000}{3600}\right)\frac{1}{s} = 279\frac{1}{s}. \tag{3.9}$$

Die Last des Wurzel-RepServers für TRS-M beträgt pro Sekunde:

$$D = \left(\frac{10000}{86400} + \frac{1000000 \cdot 2(\log_{10}(100(10-1)+1)-1)10}{86400} + \frac{1000000}{3600}\right)\frac{1}{s} \approx 323\frac{1}{s}. \tag{3.10}$$

Den größten Einfluss auf das Ergebnis haben, sowohl bei TRS-K als auch bei TRS-M, die Aktualisierungen der Gruppenregister.

Obwohl das angenommene Szenario von einem umfangreichen Einsatz zuverlässiger Gruppenkommunikation ausgeht, fallen nur 279 bzw. 323 zu verarbeitende Anfragen pro Sekunde an. Diese sind für einen modernen Rechner keine große Herausforderung, zumal diese Operationen ausschließlich im Hauptspeicher ablaufen und nicht auf langsamen stabilen Speicher angewiesen sind.

Die Aufgaben eines RepServers können mit einem Netzwerk-Router verglichen werden. Auf eine ankommende Nachricht muss möglichst schnell das passende Gruppenregister bzw. der passende Eintrag der Routing-Tabelle gefunden werden. Eine effiziente Organisation in einer Hash-Tabelle ist eine wesentliche Voraussetzung für einen hohen Durchsatz. Ist das passende Gruppenregister gefunden, muss im Regelfall lediglich die Ablaufzeit oder der Metrikbetrag angepasst, d.h. wenige Byte neu in den Hauptspeicher geschrieben oder lediglich gelesen werden. Effiziente Routingsoftware kann auf üblichen PCs durchaus mehrere 10000 Pakete pro Sekunde weiterleiten. Decasper et al. beschreiben Messergebnisse von ungefähr 30000 Paketen pro Sekunde auf einem Pentium Pro mit 233 MHz [Decasper et al. 1998]. In [Novell 1992] wird von ungefähr 5000 Paketen pro Sekunde auf Intel 486 Hardware mit 33 MHz gesprochen. Aus diesen Durchsatzangaben wird ersichtlich, dass mit heutigen Rechnern in der Gigaherzklasse und mit einer effizienten Implementierung, die oben ermittelten 279 bzw. 323 Anfragen pro Sekunde problemlos bearbeitet werden können.[3]

Sollte der Durchsatz der Such-RepServer dennoch einen Flaschenhals darstellen, kann darauf reagiert werden. Die einfachste Maßnahme stellen geographisch eingeschränkte Multicast-Gruppen dar. Diese können aufgrund ihres Inhalts oder ihrer Sprache geographisch begrenzt sein. Vor allem geschlossene Gruppen müssen in die Überlegungen miteinbezogen werden. Diese stehen der Öffentlichkeit nicht zur Verfügung, sondern können z.B. nur innerhalb eines Konzerns gegebenenfalls mit mehreren Standorten genutzt werden. Hat eine Multicast-Gruppe nur einen eingeschränkten geographischen Verbreitungsbereich, so muss ihre Existenz meist nicht bis zum Wurzel-RepServer propagiert werden, sondern nur bis zum ersten RepServer, dessen Domäne den gesamten Verbreitungsbereich einschließt. Vor allem durch die damit obsoleten Aktualisierungsnachrichten können RepServer deutlich entlastet werden.

Sollten geographisch beschränkte Multicast-Gruppen nicht ausreichen bzw. deren Einführung nicht praktikabel sein, um einen möglichen Flaschenhals zu beheben, können die Such-RepServer auch repliziert werden. Dazu wird der zu verwaltende Bereich der Multicast-Gruppen anhand der Multicast-Adresse in $n$ disjunkte Teilbereiche aufgeteilt. Ein logischer RepServer wird ebenfalls auf $n$ physische RepServer aufgeteilt, indem jeder RepServer einen Teilbereich der Multicast-Gruppen verwaltet. Ist

[3] Der Token-Repository-Service wurde zu Demonstrationszwecken in Java implementiert [Walz 1999]. Da aufgrund der hier erfolgten Analyse und Argumentation keine Leistungsengpässe zu erwarten sind lag der Schwerpunkt nicht auf einer effizienten Implementierung. Deshalb wurde auf eigene Messungen verzichtet.

ein RepServer repliziert, werden dessen Kind– und Vater-RepServer die Teilbereiche der Multicast-Gruppen und die dazugehörigen physischen RepServer mitgeteilt. Eine Gruppenerstellung, Markensuche oder Gruppenregisteraktualisierung kann folglich direkt an den zuständigen physischen RepServer zugestellt werden. Mit diesem einfachen Verfahren kann der Durchsatz proportional zur Anzahl der eingesetzten physischen RepServer erhöht werden. Abschließend lässt sich feststellen, dass einem praktischen Einsatz des Token-Repository-Service keine Speicherverbrauchs– oder Durchsatzprobleme entgegenstehen.

## 3.9 Vergleich des Token-Repository-Service mit ERS und ERA

Für einen ersten Vergleich der drei TRS-Strategien mit ERS und ERA wird im Folgenden eine Analyse beschrieben. Berücksichtigung finden dabei der maximale globale Nachrichtenaufwand und die maximale Höhe des erstellten Kontrollbaums. Diese können Anhaltspunkte dafür liefern, wie das Verhalten der unterschiedlichen Verfahren im denkbar ungünstigsten Szenario sein wird. Der durchschnittliche Nachrichtenaufwand und die durchschnittliche Höhe des erstellten Kontrollbaums wird nicht berechnet, sondern in Simulationen in Abschnitt 3.10 ermittelt. Dabei wird sich zeigen, dass in manchen Situationen die berechneten Maximalwerte tatsächlich annähernd erreicht werden können.

### 3.9.1 Nachrichtenaufwand

Für die Berechnung des Nachrichtenaufwands wird lediglich die Transportschicht für das Gruppenmanagement betrachtet und nicht die Vermittlungsschicht oder der Nachrichtenaufwand für die zuverlässige Multicast-Auslieferung. Zur Vereinfachung der Berechnung wird die Annahme getroffen, dass der erste Gruppenaustritt erfolgt, nachdem alle Gruppenbeitritte durchgeführt wurden. Auch wird nicht der Aufwand betrachtet der entsteht, wenn Gruppenführer den Kontrollbaum verlassen, da dieser maßgeblich durch das Multicast-Transportprotokoll beeinflusst wird. Obwohl in der Realität Gruppeneintritte nebenläufig erfolgen können, wird für die Berechnung strenge Sequentialität angenommen, d.h. auch die Aktualisierungen von Markencontainer und Gruppenregister sollen vor dem nächsten Eintrittswunsch abgeschlossen sein. Die Berechnung erfolgt ohne Berücksichtigung von Knoten– und Kommunikationsfehlern.

Der Nachrichtenaufwand von ERA hängt wesentlich von der betrachteten Zeitperiode ab und nicht von der Anzahl eintretender Gruppenmitglieder. Demzufolge kann ein sinnvoller Vergleich des Nachrichtenaufwands mit ERA nicht erfolgen. In Abschnitt 3.10 werden Simulationsergebnisse präsentiert, um einen Vergleich mit ERA zu erhalten.

| | | |
|---|---|---|
| $n_x$ | - | Anzahl der Nachrichten zum Erstellen einer Gruppe (x=c), Löschen einer Gruppe (x=d), Beitreten einer Gruppe (x=j) und Verlassen einer Gruppe (x=l). |
| $c, d, j, l$ | - | Anzahl durchgeführter Gruppenerstellungen, Gruppenauflösungen, Gruppenbeitritte und Gruppenaustritte. |
| $R$ | - | Anzahl der Teilnehmer pro Multicast-Gruppe. |
| $L$ | - | Anzahl an (Blatt-) RepServern bei denen die Lieferung einer Marke angefragt wird. |
| $B$ | - | Verzweigungsgrad des Kontrollbaums. |
| $K$ | - | Verzweigungsgrad der TRS-Hierarchie. |
| $h$ | - | Höhe des erstellten Kontrollbaums. |
| $t$ | - | Höhe der RepServer-Hierarchie. |
| $M$ | - | Faktor zur Gewichtung von Multicast– gegenüber Unicast-Nachrichten. |

*Tabelle 3.5: Notation für die Analyse der Nachrichtenanzahl und der Kontrollbaumhöhe*

#### 3.9.1.1 ERS

Bei ERS werden keine expliziten Gruppenoperationen zum Erstellen einer Gruppe, Löschen einer Gruppe und Verlassen einer Gruppe durchgeführt. Deren Nachrichtenaufwand ist deshalb null. Der aufwändigste Fall für den Gruppenbeitritt besteht darin, dass 255 Multicast-Suchnachrichten versendet werden müssen, bis ein Vaterknoten im Kontrollbaum gefunden wird. Die Anzahl von 255 Versuchen resultiert aus dem maximalen Teilstreckenwert (TTL) in IP-Paketen von ebenfalls 255. Der schlechteste Fall für die Anzahl der Antworten von Kontrollbaummitgliedern besteht darin, dass anschließend alle Kontrollbaummitglieder dem suchenden Knoten antworten. Die maximale Anzahl Suchnachrichten $n_j$ für den Gruppenbeitritt ist daher wie folgt:

$$\begin{aligned} n_j &= \mathit{Suchnachrichten} + \mathit{Antworten} \\ &= 255\,jM + \sum_{i=1}^{j} i \\ &= 255\,jM + \frac{j^2 + j}{2}. \end{aligned} \tag{3.11}$$

$M$ ist ein Faktor zur Höhergewichtung von Multicast– gegenüber Unicast-Nachrichten und $j$ ist die Anzahl an Gruppenbeitritten (siehe Tabelle 3.5). Dadurch, dass die Anzahl der Gruppenbeitritte quadratisch in das Ergebnis mit eingeht, steigt auch der maximale Nachrichtenaufwand quadratisch an, was offensichtlich ein ungünstiges Verhalten darstellt.

#### 3.9.1.2 TRS-R

Der Nachrichtenaufwand für TRS-R zum Anlegen einer Gruppe ist exakt eine Nachricht, die *repCreateGroup*-Nachricht, die an den lokalen RepServer gesendet wird. Zum Löschen einer Gruppe

wird *repDeleteGroup* an den lokalen RepServer gesendet, der eine Multicast-Nachricht an alle RepServer sendet, d.h.:

$$n_d = d + dM. \tag{3.12}$$

Der Gruppenbeitrittswunsch wird an den lokalen RepServer gesendet, der mit einer Marke antwortet. Ist hingegen keine Marke vorhanden, wird eine Markensuche eingeleitet, indem Multicast-Suchnachrichten versendet werden. Auch hier besteht der denkbar schlechteste Fall aus 255 Suchnachrichten gefolgt von Antworten aller anderen RepServer. Schließlich wird noch die Marke übergeben, das weitere zwei Nachrichten benötigt. Im Gegensatz zu ERS wird eine derartige Suche indessen nur maximal einmal pro Gruppe und RepServer durchgeführt. Nachfolgende Eingliederungen in den Kontrollbaum können durch lokale Marken erfolgen. Der RepServer, der die Gruppe erstellt hat, benötigt keine Markensuche. Der maximale Nachrichtenaufwand ergibt sich schließlich wie folgt, wobei $L$ die Anzahl abgefragter RepServer kennzeichnet:

$$\begin{aligned} n_j &= 2j + (L-1)(255M + (L-1) + 2) \\ &= 2j + 255M(L-1) + L^2 - 1. \end{aligned} \tag{3.13}$$

Diese Formel enthält ebenfalls eine quadratische Komponente. Sie bezieht sich im Gegensatz zu ERS jedoch auf die Anzahl von RepServern und nicht auf die Anzahl der Gruppenbeitritte, die wesentlich höher ausfallen kann. Unter der Annahme, dass die Anzahl der Gruppenbeitritte wesentlich größer ist als die Anzahl der RepServer ergibt sich folgende Näherung:

$$n_j \approx 2j, \tag{3.14}$$

d.h. eine linear steigende Nachrichtenanzahl bezogen auf die Anzahl von Gruppenbeitritten.

Das Verlassen einer Gruppe ist wiederum recht einfach. Es wird dazu eine Nachricht von dem verlassenden Mitglied an den lokalen RepServer gesendet und eine *repAddToken*-Nachricht des Vaterknotens im Kontrollbaum an dessen RepServer.

#### 3.9.1.3 TRS-K

Um eine Gruppe zu erstellen ist bei TRS-K nicht nur eine Nachricht an den lokalen RepServer notwendig, sondern auch das Anlegen von Gruppenregistern auf allen RepServern zwischen dem lokalen RepServer und der Wurzel. Dazu sind *t-1* Nachrichten erforderlich, wobei $t$ die Höhe der TRS-Hierarchie ist, gezählt mit der Wurzel auf Höhe eins:

$$n_c = t. \tag{3.15}$$

Zum Löschen einer Gruppe wird eine Nachricht an den lokalen RepServer gesendet und von dort entlang des Gruppenbaums, als Teilbaum der TRS-Hierarchie, weitergeleitet. Die TRS-Hierarchie hat den maximalen Verzweigungsgrad $K$. Die Nachrichtenanzahl dafür ist:

$$n_d = \sum_{i=0}^{t-1} K^i. \tag{3.16}$$

In der Formel wird die Anzahl der Kanten in der TRS-Hierarchie gezählt inklusive der zusätzlichen Nachricht des Multicast-Teilnehmers an seinen lokalen RepServer.

Beim Gruppenbeitritt findet eine Unterscheidung zwischen den ersten *L-1* Gruppenbeitritten und allen folgenden statt. Die ersten *L-1* Gruppenbeitritte sollen an verschiedenen RepServern erfolgen und folglich jedesmal eine Markensuche bedingen. Im ungünstigsten Fall muss die Markensuche bis zum Wurzel-RepServer weitergeleitet werden und von dort aus zu einem Blatt-RepServer. Dies erfolgt mit $2(t-1)$ Nachrichten. Wird die gefundene Marke übergeben, muss der Gruppenbaum aktualisiert werden, was wiederum mit $t-1$ Nachrichten erfolgt. Drei einzelne Nachrichten sind erforderlich für die *repJoinGroup*-Nachricht, die Übergabe der gefundenen Marke an den suchenden RepServer und schließlich die Übergabe der Marke an das neue Mitglied. Nach den ersten *L-1* Gruppenbeitritten sind keine weiteren Markensuchen mehr erforderlich, da alle RepServer aus einer bereits vorangegangenen Markensuche lokale Marken besitzen. Somit sind nur noch zwei Nachrichten pro Gruppenbeitritt erforderlich. Die Gesamtzahl erforderlicher Nachrichten beträgt:

$$n_j = \sum_{i=1}^{t-1}((K^i - K^{i-1})(3(t-i)+3)) + 2(j-L+1), \qquad j > L. \tag{3.17}$$

Verlässt ein Mitglied den Kontrollbaum, kann dies eine Aktualisierung des Gruppenbaums nach sich ziehen. Daraus folgt, dass im aufwändigsten Fall *t+1* Nachrichten für den Gruppenaustritt benötigt werden, berücksichtigt man noch die *repAddToken*-Nachricht des Vaterknotens im Kontrollbaum.

#### 3.9.1.4 TRS-M

Die Analyse von TRS-M ist, aufbauend auf der vorangegangenen Analyse von TRS-K, unkompliziert. Die Nachrichtenanzahl für Gruppenerstellung, Gruppenauflösung und Gruppenaustritt ist gegenüber TRS-K unverändert. Im schlechtesten Fall resultiert jeder Gruppenbeitritt in einer globalen Markensuche und in einer Aktualisierung der Gruppenregister. Der Nachrichtenaufwand dafür beträgt:

$$n_j = j(3(t-1)+3). \tag{3.18}$$

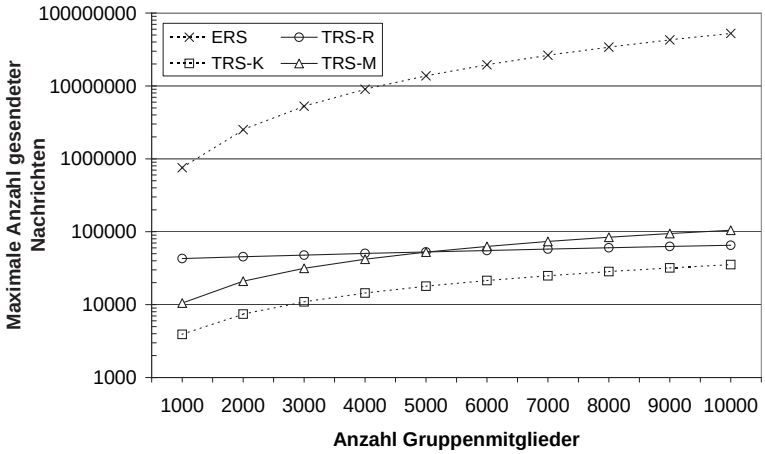



*Abbildung 3.28: Maximale Nachrichtenanzahl für den Kontrollbaumaufbau mit ERS und TRS*

#### 3.9.1.5 Numerische Ergebnisse des Nachrichtenaufwands

In Abbildung 3.28 ist die maximale Anzahl an Nachrichten der verschiedenen Protokolle für den Kontrollbaumaufbau dargestellt. Abgebildet ist das Resultat für ein Szenario, in dem eine Gruppe angelegt wird, die auf der x-Achse angegebene Anzahl von Mitgliedern der Gruppe beitritt und die Hälfte davon die Gruppe auch wieder verlässt. Für die TRS-Protokolle werden 111 RepServer angenommen, die für TRS-K und TRS-M in einem TRS-Baum mit Höhe drei und Verzweigungsgrad zehn (=$K$) angeordnet sind, was in 100 Blatt-RepServern resultiert (=$L$).

Das Ergebnis zeigt, dass ERS nicht geeignet ist, um den Kontrollbaumaufbau für große Gruppen zu übernehmen. Zu beachten ist, dass die y-Achse logarithmisch skaliert ist. Neben der um mehrere Größenordnungen höheren Nachrichtenanzahl von ERS im Vergleich zu TRS, muss nochmals betont werden, dass dies lediglich der Aufwand auf Transportschicht ist und auch Multicast-Nachrichten beinhaltet. Auf der Vermittlungsschicht ist für ERS nochmals ein hoher Aufwand einzukalkulieren, wie es in Abschnitt 3.2.3 dargelegt wurde. Demgegenüber erzeugt TRS einen sehr geringen Aufwand, der zudem eine gute Skalierbarkeit, bezogen auf die Anzahl der Gruppenbeitritte, aufweist. Das Protokoll TRS-K resultiert im geringsten Nachrichtenaufwand. Der Nachrichtenaufwand für TRS-M liegt etwas über dem Aufwand von TRS-R. Jedoch muss bedacht werden, dass TRS-M nur Unicast-Nachrichten verwendet, während in TRS-R analog zu ERS auch Multicast-Nachrichten eingesetzt werden, die ebenfalls jeweils nur als eine Nachricht in die Zählung eingehen. Außerdem stellt der Wert für TRS-M die Obergrenze für äußerst ungünstige Anmeldungen mit prinzipiell notwendiger Markensuche dar und es erfolgte zudem keine Berücksichtigung der wählbaren Toleranz bei der Markensuche, was die Nachrichtenanzahl senkt.

### 3.9.2 Kontrollbaumhöhe

Die Kontrollbaumhöhe beeinflusst die Zuverlässigkeit des Kontrollbaums sowie die Umlaufverzögerung, die z.B. für potentielle Übertragungswiederholungen, aber auch für den Durchsatz bei einem fensterbasierten Sendemechanismus und die Speicherfreigabe von Bedeutung ist (siehe Abschnitt 3.5.2 und 4.6.4). Deshalb sind niedrige Baumhöhen erstrebenswert.

#### 3.9.2.1 ERS und ERA

Mit ERS und ERA ist die Kontrollbaumhöhe lediglich durch die Anzahl an Gruppenmitgliedern limitiert. Dies bedeutet, dass im schlechtesten Fall die Kontrollbaumhöhe gleich $j-l+1$ ist, wobei $j$ die Anzahl der Gruppeneintritte und $l$ die Anzahl der Gruppenaustritte ist.

#### 3.9.2.2 TRS-R, TRS-K und TRS-M

Bei TRS-R und TRS-K hat nicht nur die Anzahl der Gruppenmitglieder Einfluss auf die Kontrollbaumhöhe, sondern auch die Anzahl abgefragter RepServer. Wird nur ein RepServer abgefragt, d.h. gehen alle *repJoinGroup*-Operationen an den selben RepServer, wird ein Kontrollbaum mit minimaler Höhe erzeugt, da bei jeder Markenwahl eine Marke mit dem niedrigsten Metrikbetrag gewählt wird. Die Anzahl Empfänger $R$ in einem balancierten Kontrollbaum der Höhe $h$ ist:

$$\begin{aligned} R &= \sum_{i=0}^{h-1} B^i = B^0 + B^1 + \ldots + B^{h-2} + B^{h-1} \\ &= \frac{(1-B)B^0}{1-B} + \frac{(1-B)B^1}{1-B} + \ldots + \frac{(1-B)B^{h-2}}{1-B} + \frac{(1-B)B^{h-1}}{1-B} \\ &= \frac{B^0 - B^1 + B^1 - B^2 + \ldots + B^{h-2} - B^{h-1} + B^{h-1} - B^h}{1-B} \\ &= \frac{1-B^h}{1-B}. \end{aligned} \tag{3.19}$$

Folglich ist die Kontrollbaumhöhe $h$:

$$h = \log_B(R(B-1)+1). \tag{3.20}$$

Die Kontrollbaumhöhe mit $j$ Gruppenbeitritten ist damit:

$$h = \lceil \log_B((j+1)(B-1)+1) \rceil. \tag{3.21}$$

Nun müssen noch alle $L$ lokalen RepServer berücksichtigt werden, von denen eine Marke abgefragt wird. Im schlechtesten Fall wird an $L-1$ RepServern nur jeweils eine Marke abgefragt und jede

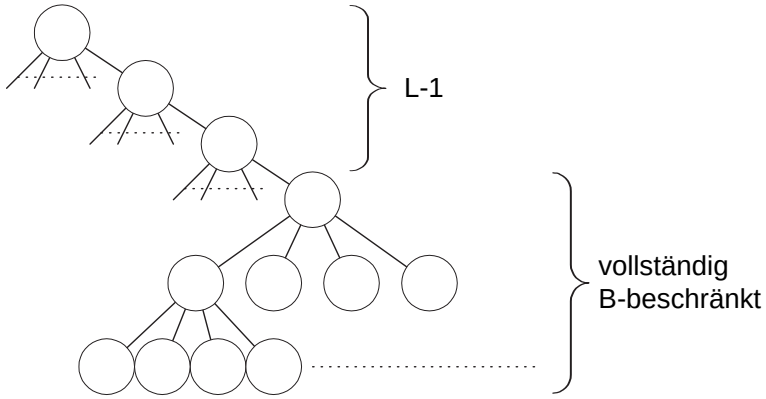



*Abbildung 3.29: Berechnung der Kontrollbaumhöhe*

Markenabfrage liefert eine Marke mit maximaler Höhe im Kontrollbaum. Alle anderen Gruppeneintritte werden von einem RepServer durchgeführt. Dadurch kann der Kontrollbaum durch die $L-1$ RepServer die Höhe $L-1$ erhalten und durch den letzten RepServer die Höhe eines vollständig ausgeglichenen Baums mit Verzweigungsgrad $B$ (siehe Abbildung 3.29). Damit ist die maximale Kontrollbaumhöhe:

$$h = L-1+\lceil \log_B((j-l-L+2)(B-1)+1)\rceil. \tag{3.22}$$

Für TRS-M ergibt sich die Kontrollbaumhöhe aus einem vollständig ausgeglichenen Baum zu:

$$h = \lceil \log_B((j-l+1)(B-1)+1)\rceil. \tag{3.23}$$

#### 3.9.2.3 Numerische Ergebnisse der Kontrollbaumhöhe

In Abbildung 3.30 ist die maximale Höhe entstehender Kontrollbäume für die unterschiedlichen Protokolle dargestellt. Angenommen wurde ein Szenario, in dem zwischen 1000 und 10000 Gruppenbeitritte erfolgen und ebenfalls wie im Szenario für die Nachrichtenanzahl aus dem vorigen Abschnitt, 111 RepServer betroffen sind.

Das Ergebnis für TRS-M entspricht der minimalen Baumhöhe für vollständig verzweigte Bäume. TRS-R und TRS-K liefern Kontrollbäume mit mittleren Höhen. Die Kontrollbaumhöhe bleibt für den dargestellten Bereich der Gruppenbeitritte praktisch konstant, da der bestimmende Parameter die konstante Anzahl der RepServer ist. ERS und ERA können in sehr großen Kontrollbaumhöhen und dementsprechend ungünstigen Kontrollbäumen resultieren.

## 3.10 Simulationen

Das Ziel dieses Abschnitts ist es die Analyseergebnisse und bisher nur argumentativ vorgetragene Behauptungen durch Simulationsergebnisse zu belegen. Simulationen wurden dabei Messungen in realen Umgebungen vorgezogen, da sich diese nicht mit einem akzeptablen Aufwand hinsichtlich der notwendigen großen Teilnehmerzahlen realisieren lassen.

Alle Simulationen wurden mit dem frei verfügbaren und erweiterbaren Simulationssystem NS-2 durchgeführt [Bajaj et al. 1999]. NS-2 bietet mit den Routing-Protokollen DVMRP [Waitzman et al. 1988] und PIM-SM [Estrin et al. 1998] bereits eine geeignete Multicast-Unterstützung. Eigene Implementierungen mussten für ERS, ERA, die TRS-Protokolle sowie später in Abschnitt 4.7 für die Transportprotokolle erfolgen. Desweiteren wurde ein Generator für Nutz– und Hintergrundlast implementiert. Um realistische Netzwerke für eine Simulation zu erhalten, werden vielfach die beiden Netzwerkgeneratoren Tiers [Calvert et al. 1997] und GT-ITM [Zegura et al. 1997] eingesetzt. Tiers erzeugt streng hierarchische Netzwerke, während GT-ITM zusätzlich direkte Verbindungen zwischen beliebigen Knoten erzeugt. Für die nachfolgenden Simulationen wurden beide Netzwerkgeneratoren eingesetzt. Eine Diskussion derer findet sich in [Böser 1999].

Alle Simulationen wurden auf einer Sun Enterprise E4500/E5500 mit zwölf 400 MHz Prozessoren und zwölf GByte Hauptspeicher durchgeführt. Die verwendete Hardware hat keinen Einfluss auf die Ergebnisse, allerdings können mit leistungsfähigeren Rechnern auch aufwändigere Szenarien simuliert werden. Bei der Arbeit mit NS-2 hat leider dessen begrenzte Skalierbarkeit des öfteren eine unfreiwillige Grenze für die Versuche gesetzt. So haben sich bei einem großen Nachrichtenaufkommen, z.B. bedingt durch künstlich eingeführte Hintergrundlast, die Simulationszeiten unangenehm in die Länge gezogen. Mitunter dauerte ein Simulationsdurchlauf mehrere Tage. Daher wurden die meisten

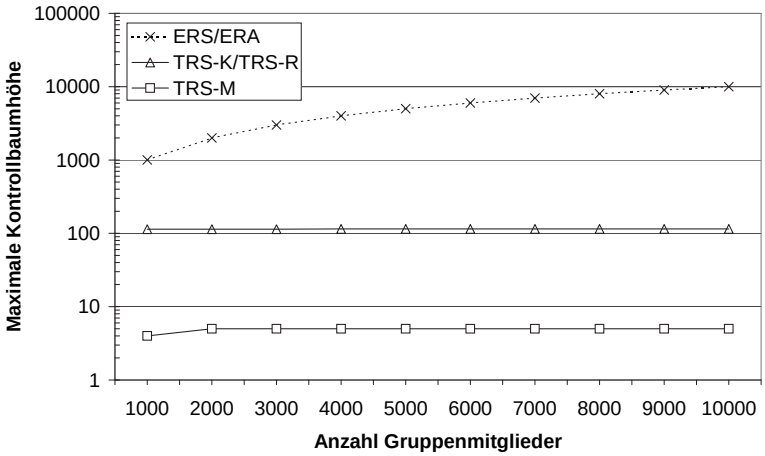



*Abbildung 3.30: Maximal entstehende Kontrollbaumhöhe mit ERS, ERA und TRS*

Simulationen ohne Hintergrundlast ausgeführt. Auch der Netzwerkgröße wurde durch einen überproportionalen Anstieg des Speicherverbrauchs eine Grenze gesetzt. Diese Grenze lag bei ungefähr 2000 Knoten. Wiederum wurden für die meisten Simulationen dennoch kleinere Netzwerkgrößen gewählt, um den Zeitbedarf in akzeptablem Umfang zu halten.

Die Ausführungen zur Simulation beginnen mit einer Beschreibung der Testumgebung. Daran anschließend werden die Simulationsresultate illustriert und diskutiert.

### 3.10.1 Testumgebung

Die erzeugten Netzwerke haben zwischen 100 und fast 2000 Knoten. Die Bandbreite der Verbindungen beträgt zwischen 10 MBit/s und 1000 MBit/s. Die zufällig gewählten Verzögerungen pro Verbindung betragen zwischen 1 und 3 Millisekunden für LANs, zwischen 1 und 8 Millisekunden für MANs und zwischen 5 und 19 Millisekunden für WANs.[4] Die absoluten Werte für Bandbreite und Verzögerung beeinflussen lediglich die quantitativen Resultate, nicht aber die qualitativen, was durch Kontrollmessungen bestätigt wurde.

Versuche wurden sowohl mit DVMRP– als auch mit PIM-SM-Routing durchgeführt. Die Routing-Tabellen wurden statisch, vor Beginn der Simulation, festgelegt. Obwohl dies NS-2 unterstützt, wurde keine dynamische Anpassung der Routing-Pfade an die Lastsituation durchgeführt. Begründet ist dies durch die nicht akzeptablen Ausführzeiten einer dynamischen Simulation, die um Größenordnungen über den Ausführzeiten von statischem Routing liegt. Die gegebenenfalls beaufschlagte Hintergrundlast wurde durch zufällig platzierte Sender und Empfänger von TCP-Verkehr erzeugt. Die erzeugte Last eines einzelnen TCP-Senders ist exponentialverteilt, mit einer Sendezeit von einer Sekunde und einer Wartezeit von fünf Sekunden. Alle Nachrichten wurden zuverlässig zugestellt. In allen Simulationen wurde der maximale Verzweigungsgrad des entstehenden Kontrollbaums auf zehn Kindknoten pro Gruppenführer beschränkt.

Der TRS-Dienst ist mit acht Blatt-RepServern konfiguriert. Die hierarchischen Varianten des TRS-Diensts, TRS-K und TRS-M, wurden mit einem Verzweigungsgrad von zwei versehen. Dies resultierte in insgesamt 15 RepServern. In Kontrollmessungen wurden außerdem davon abweichende Konfigurationen des TRS-Diensts mit mehr oder weniger RepServern sowie unterschiedlichen Verzweigungsgraden und Höhen der TRS-Hierarchie untersucht. Periodische Markenaktualisierungen werden nicht betrachtet und sind darüber hinaus auch nicht notwendig, da keine Knotenausfälle simuliert werden.

[4] Aus Messungen in [Carter & Crovella 1996] und [Theilmann 2000] lässt sich schließen, dass die durchschnittliche Verzögerung pro Teilstrecke im Internet im Bereich von ungefähr 8 Millisekunden liegt. Eine genauere Unterscheidung zwischen LAN–, MAN– und WAN–Strecken wurde nicht durchgeführt. Die hier verwendeten Werte liegen im Bereich dieser Messungen.

Bis auf anderslautende Beschreibungen zu den Simulationsresultaten wurde ERS mit einer Wartezeit bis zum erneuten Senden einer Suchnachricht von 0,5 Sekunden pro Teilstrecke konfiguriert, was für die simulierten Netzwerke im unbelasteten Fall eine ausreichende Wartezeit ist.[5] Für den belasteten Fall werden im nächsten Abschnitt detailliertere Erläuterungen zu den Wartezeiten erfolgen. Dies stellt einen akzeptablen Kompromiss zwischen der Wartezeit für die Eingliederung und dem Nachrichtenaufkommen dar. Ein suchender Knoten wählt den ersten positiv antwortenden Knoten zu seinem Vaterknoten im Kontrollbaum. Dieser weist für gewöhnlich die geringste Netzwerkdistanz auf. Bei gleichwertiger Netzwerkdistanz mehrerer Knoten wird durch dieses Verfahren derjenige mit der geringsten Verzögerung gewählt. Für ERA wurde eine Wartezeit zwischen zwei aufeinanderfolgenden Suchnachrichten von 20 Sekunden konfiguriert. Entsprechend der ursprünglichen Verwendung von ERA in [Paul et al. 1997] wird der Ausbreitungsbereich der Suchnachrichten nicht schrittweise erhöht, sondern bleibt auf einem festen, für das Netzwerk ausreichend großen Wert. Ein neuer Teilnehmer wartet mindestens zehn Sekunden auf das Eintreffen von Suchnachrichten.[6] Stehen mehrere Vaterknoten zur Auswahl, wird derjenige mit der geringsten Netzwerkdistanz gewählt. Ist die Wartezeit abgelaufen, ohne dass mehrere Suchnachrichten eingetroffen sind, wird der Absender der ersten eintreffenden Suchnachricht zum Vaterknoten gewählt. Die Konfiguration von TRS-R entspricht weitgehend der von ERS, d.h. Suchnachrichten werden mit ansteigendem TTL versendet und die Wartezeit beträgt ebenfalls 0,5 Sekunden pro Teilstrecke.

## 3.10.2 Simulationsresultate

### 3.10.2.1 Nachrichtenaufwand in Abhängigkeit von der Hintergrundlast

Die ersten Resultate betreffen die äußerst wichtige Skalierbarkeit der verschiedenen Protokolle zum Aufbau von Kontrollbäumen. Die Simulation basiert auf einem mit Tiers erzeugten Netzwerk bestehend aus 251 Knoten. Simuliert wurde eine Zeitspanne von 200 Sekunden, in denen 200 Anmeldungen an eine neue Gruppe durchgeführt wurden. Als Routing-Protokoll kam DVMRP zum Einsatz. Um eine hohe Auslastung der Verbindungen bei gleichzeitig akzeptabler Simulationsdauer zu erreichen, wurden in dieser Simulation die Bandbreiten der Verbindungen um den Faktor 100 erniedrigt. Sie

[5] In der Literatur lassen sich keine Vorschläge für die Wartezeit zwischen zwei Suchnachrichten finden. Die hier gewählte Konfiguration liegt deutlich über der Umlaufverzögerung einer Teilstrecke, so dass auch bei einem moderat belasteten Netzwerk genügend lange auf Antworten gewartet wird.

[6] In [Paul et al. 1997] ist die Wartezeit zwischen zwei Suchnachrichten von Gruppenführern mit 3 Sekunden konfiguriert, allerdings war die Gruppengröße mit 18 Empfängern deutlich kleiner. Für die Wartezeit eines neuen Teilnehmers bis zur Auswahl eines Gruppenführers wurden keine Angaben gemacht. Für die folgenden Simulationen wurde die Gruppenführerwartezeit deutlich größer gewählt, da die Netzbelastung ansonsten enorm gewesen wäre und ERA dementsprechend schlecht abgeschnitten hätte.

betrugen nun zwischen 100 KBit und 10 MBit. Die parametrisierbare Hintergrundlast wurde durch eine Senderate der TCP-Sender zwischen null und zwölf KBit/s erzeugt.

In Abbildung 3.31 ist die Gesamtanzahl empfangener Nachrichten für den Kontrollbaumaufbau in Abhängigkeit von der Hintergrundlast aufgezeigt. Für ERS und ERA werden diese Nachrichten von den Teilnehmern der zuverlässigen Multicast-Gruppe empfangen und bearbeitet sowie von den neuen Teilnehmern vor dem Eintritt. Für die TRS-Protokolle sind es zusätzlich die RepServer, die Nachrichten empfangen. Die dargestellte Gesamtanzahl ist die Summe der empfangenen Nachrichten über alle Teilnehmer, neue Teilnehmer und gegebenenfalls RepServer und entspricht der verursachten Last im Netzwerk zum Kontrollbaumaufbau für 200 Gruppenmitglieder. Es werden hier die empfangenen Nachrichten anstelle der gesendeten gezählt, da ERS, ERA und TRS-R bei der Gruppenführersuche auf einen Multicast-Dienst zurückgreifen. Dadurch würde die Vergleichbarkeit zwischen einer gesendeten Multicast-Nachricht an potentiell viele Empfänger und einer gesendeten Unicast-Nachricht erschwert.

Die Teilabbildung a) zeigt die Ergebnisse für ERS mit verschiedenen Wartezeiten. Die Wartezeit bestimmt, ab wann eine gesendete und unbeantwortete Suchnachricht als nicht erfolgreich angesehen wird und eine neue mit erhöhtem TTL gesendet wird. Zu beachten gilt, dass diese Zeit mit dem TTL der gesendeten Suchnachricht multipliziert werden muss, um die tatsächliche Wartezeit für eine bestimmte Suchnachricht zu erhalten. Damit kann die Wartezeit auf die zunehmende Nachrichtenverzögerung angepasst werden. Die Hintergrundlast ist hier definiert als der prozentuale Zeitanteil vollständig ausgelasteter Verbindungen während der Simulation. Eine Hintergrundlast von 100% bedeutet somit, dass während der gesamten Simulation alle Verbindungen vollständig ausgelastet waren. Eine Verbindung ist zu einem Zeitpunkt ausgelastet, wenn die Sendewarteschlange nicht leer ist. Die Abbildung zeigt eine Hintergrundlast bis zu 19%. Dies erscheint wenig, doch muss beachtet werden, dass eine Auslastung von 100% jede einzelne Verbindung betreffen muss und darum recht unwahrscheinlich ist. Bereits die 19%-Auslastung hat die stark frequentierten und folglich wichtigen Verbindungen ausreichend belastet.

Die Abbildung zeigt, dass ERS ein sehr ungünstiges Verhalten mit steigender Hintergrundlast aufweist. Exemplarisch soll ERS mit 0,1 Sekunden Wartezeit pro Teilstrecke betrachtet werden. Angenommen, es muss bis zu einem TTL von zehn erhöht werden, dann bedeutet dies für das neue Gruppenmitglied eine Wartezeit von $0,1 + 0,2 + 0,3 + \ldots + 0,9 + 1,0 = 5,5$ Sekunden, bevor ein Vaterknoten im Kontrollbaum gefunden wird. Obwohl diese Wartezeit nicht besonders niedrig erscheint führt sie doch zu einem äußerst hohen Verkehrsaufkommen, das durch die ERS Such– und Antwortnachrichten ausgelöst wird. Verursacht wird dieser Effekt durch die steigende Nachrichtenverzögerung mit zunehmender Hintergrundlast. Der mit ERS suchende Knoten kann in dessen Folge zu frühzeitig die Entscheidung treffen, eine neue Suchnachricht mit erhöhtem TTL auszusenden. Passiert dies mehrmals bevor die erste Antwort eintrifft, so wurde bereits an einen großen Bereich die

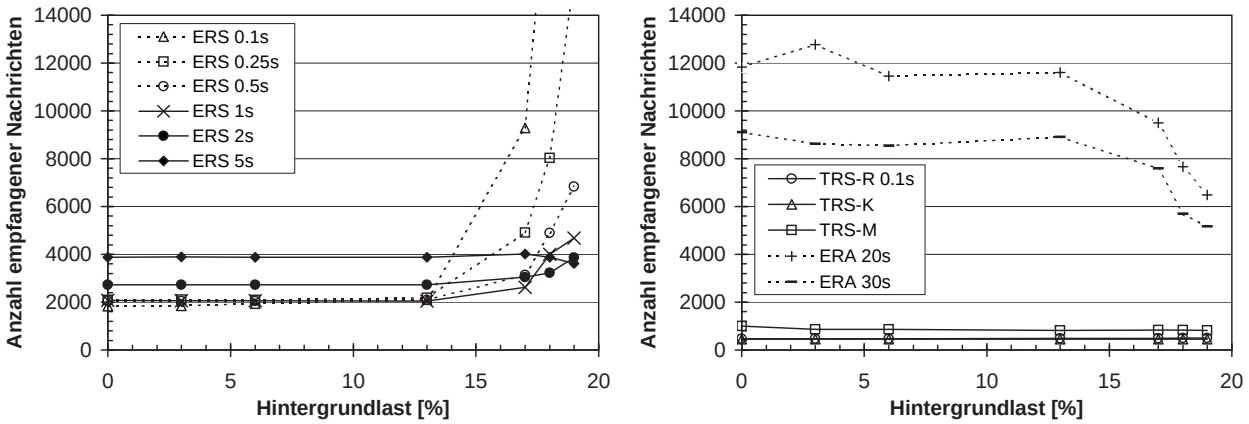



*Abbildung 3.31: Empfangene Nachrichten in Abhängigkeit von der Hintergrundlast bei a) ERS und b) ERA und TRS*

Suchnachricht gesendet und dementsprechend viele Antwortnachrichten treffen in der Folge ein. Zu beachten ist ferner, dass dieser Effekt sich selbst verstärkt, da durch die zuviel gesendeten Such- und Antwortnachrichten die Last und damit die Verzögerung im Netzwerk weiter zunimmt.

Erhöht man die Wartezeit, führt dies zu einer Abschwächung des unerwünschten Effekts. Eine Wartezeit von zwei Sekunden führt in dem betrachteten Bereich der Hintergrundlast „lediglich“ zu knapp einer Verdopplung der Nachrichtenanzahl. Für das obige Beispiel, dass ein TTL-Bereich von zehn abgesucht werden muss, bedeutet dies aber bereits eine Wartezeit von knapp zwei Minuten. Ein weiterer Nachteil wird bei einer genaueren Betrachtung der Abbildung ersichtlich. Eine erhöhte Wartezeit führt bei geringer Hintergrundlast zu einer erhöhten Nachrichtenanzahl. Wie lässt sich dieser Effekt erklären? Größere Wartezeiten führen zu größeren Verzögerungen bevor ein Mitglied in den Kontrollbaum eingegliedert werden kann. Damit stehen für einen suchenden Knoten zu einem bestimmten Zeitpunkt auch weniger Vaterknoten zur Verfügung und diese sind im Mittel vom suchenden Knoten weiter entfernt (siehe auch Diskussion in Abschnitt 3.2.3). Dies erklärt die Beobachtung der gestiegenen Nachrichtenanzahl mit erhöhter Wartezeit. Die Grafik zeigt für die Wartezeit von fünf Sekunden sogar eine Verringerung der Nachrichten bei 19% Hintergrundlast, was auf eine Verzögerung bei der Eingliederung zurückgeführt werden kann. Eine weitere Erhöhung der Hintergrundlast würde allerdings auch hier die Nachrichtenanzahl deutlich erhöhen.

Abbildung 3.31b zeigt die Resultate für ERA und TRS. ERA ist sowohl mit 20 als auch mit 30 Sekunden Wartezeit zwischen zwei Suchnachrichten gemessen worden. Beide Konfigurationen zeigen eine hohe absolute Nachrichtenanzahl. Im Gegensatz zu ERS ist bei ERA mit steigender Hintergrundlast eine absinkende Nachrichtenanzahl zu beobachten. Dieser Effekt ist lediglich durch das simulierte Szenario bedingt, das eine feste Simulationszeit von 200 Sekunden umschließt. Mit steigender Hintergrundlast und damit zunehmender Nachrichtenverzögerung können innerhalb der beschränkten

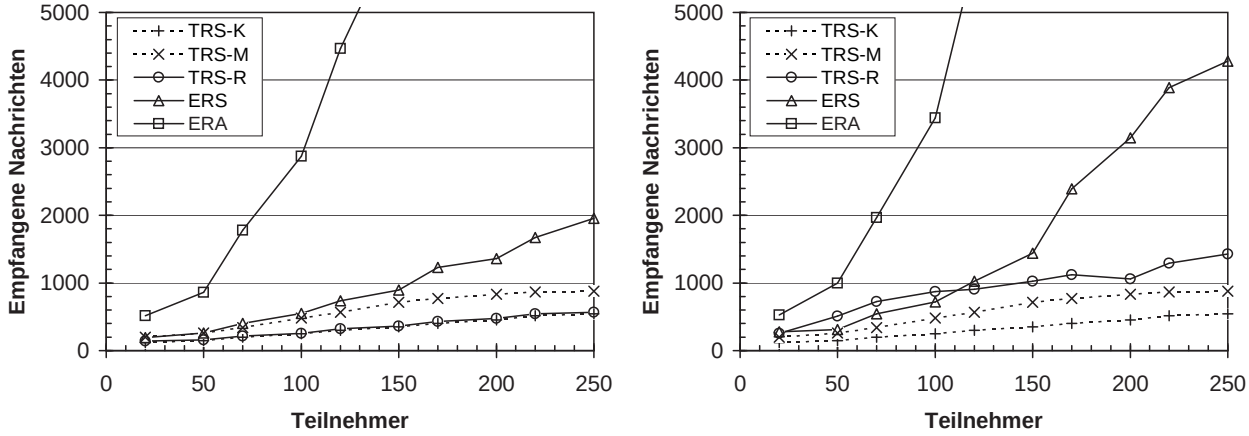



*Abbildung 3.32: Empfangene Nachrichten in Abhängigkeit von der Anzahl an Gruppenbeitritten mit a) DVMRP– und b) PIM-SM-Routing*

Simulationszeit nicht mehr alle Nachrichten zugestellt werden, was diese scheinbar sinkende Nachrichtenanzahl erklärt. In Wirklichkeit ist die Nachrichtenanzahl von ERA unabhängig von der Hintergrundlast.

Die Ergebnisse aller drei TRS-Varianten zeigen eine überaus niedrige absolute Nachrichtenanzahl und eine sehr gute Skalierbarkeit mit steigender Hintergrundlast. Selbst die Variante TRS-R, deren erweiternde Ringsuche mit konfigurierten 0,1 Sekunden eine vergleichsweise niedrige Wartezeit aufweist, zeigt in der Grafik ein konstantes Verhalten. Bei genauer Betrachtung der Zahlenwerte ist das Ergebnis nicht vollkommen konstant, sondern steigt tatsächlich leicht an, von 467 auf 507 Nachrichten innerhalb des Bereichs der simulierten Hintergrundlast. Dies stellt im Vergleich zu ERS dennoch eine beachtliche Verbesserung dar. Insgesamt erfordert TRS-K mit ungefähr 450 Nachrichten die geringste Nachrichtenanzahl der TRS-Varianten und TRS-M mit ungefähr 850 empfangenen Nachrichten die höchste Nachrichtenanzahl. Damit benötigt TRS-K nur unwesentlich mehr als die erforderliche Minimalanzahl von zwei Nachrichten (*repJoinGroup* und Lieferung der Marke) für jede der 200 Anmeldungen.

#### 3.10.2.2 Nachrichtenaufwand in Abhängigkeit von der Teilnehmerzahl

Abbildung 3.32 illustriert die Anzahl empfangener Nachrichten für Gruppenanmeldungen in Abhängigkeit von der Anzahl der Gruppenbeitritte. Für die Protokolle ERS und ERA sind es die Teilnehmer der zuverlässigen Multicast-Gruppe, neue Teilnehmer und Teilnehmer die bereits Teil des Kontrollbaums sind, die Nachrichten empfangen. Für die TRS-Protokolle sind es zusätzlich die RepServer, die Nachrichten empfangen. Die Abbildung zeigt die Gesamtzahl der empfangenen Nachrichten, die im Netzwerk zum Kontrollbaumaufbau übertragen werden und damit die Last im Netzwerk in Abhängigkeit von der Anzahl an Multicast-Teilnehmern.

In dieser und allen folgenden Simulationen wurde auf die Hintergrundlast verzichtet und die Bandbreite der Verbindungen wieder auf 10 MBit bis 1000 MBit erhöht. Erwartungskonform steigt der Aufwand bei allen Protokollen mit der Teilnehmerzahl. ERA zeigt dabei das ungünstigste Verhalten, obwohl die mit 200 Sekunden kurze Simulationszeit recht ERA-freundlich konfiguriert war. Eine längere Simulationszeit würde die Ergebnisse für ERA proportional verschlechtern, während sich für die anderen Verfahren keine Änderungen ergeben würden. Das ständige Senden von Multicast-Suchnachrichten aller noch nicht vollständig verzweigten Gruppenführer führt zu einer nicht akzeptablen Nachrichtenbelastung. Die vorgestellten Ergebnisse für ERA können durch Erhöhung der Wartezeit von 20 Sekunden zwischen zwei Suchnachrichten noch verbessert werden. Dies führt tendenziell aber auch zu größeren Distanzen und Verzögerungen zwischen Vater- und Kindknoten im Kontrollbaum, da mit höherer Wahrscheinlichkeit nicht mehr der optimale Vaterknoten ausgewählt wird. Die TRS-Varianten erzielen die geringste Nachrichtenbelastung, geringer als ERS und deutlich geringer als ERA. Wird DVMRP-Routing eingesetzt, so sind TRS-K und TRS-R mit annähernd gleichem Aufwand dem TRS-M Verfahren überlegen. Mit PIM-SM-Routing verschlechtert sich TRS-R und vor allem ERS sehr deutlich, während TRS-K und TRS-M einen identischen und ERA einen annähernd gleichen Aufwand aufweisen. Dies wird in der folgenden Abbildung ausgeprägter erkennbar sein und dort erklärt werden.

Die Anzahl gesendeter Nachrichten in Abhängigkeit von der Gruppengröße verhält sich ähnlich zu der in Abbildung 3.32 illustrierten Anzahl empfangener Nachrichten. Die absoluten Größen von gesendeten und empfangenen Nachrichten sind für TRS-K und TRS-M identisch, da kein Multicast verwendet wird. Für TRS-R, ERS und ERA kann die Anzahl gesendeter Nachrichten je nach Situation geringer oder größer als die Anzahl empfangener Nachrichten sein. Es können mehr Nachrichten empfangen werden als gesendet wurden, da mit Multicast eine Empfängergruppe adressiert wird. Andererseits können auch weniger Nachrichten empfangen werden als gesendet wurden, da die Multicast-Suchnachrichten mit einem begrenzten TTL gesendet werden und damit gegebenenfalls von keinem Knoten empfangen werden. Für das Szenario aus Abbildung 3.32 wurden beispielsweise für TRS-R ca. 20% weniger, für ERS ca. 35% mehr und für ERA ca. 90 % weniger Nachrichten gesendet als empfangen wurden. Der deutliche Unterschied bei ERA resultiert aus den globalen Multicast-Suchnachrichten, die nicht im Ausbreitungsbereich eingeschränkt werden und folglich immer von allen Multicast-Teilnehmern empfangen werden.

Der Vergleich mit den berechneten Maximalwerten aus Abschnitt 3.9 fällt folgendermaßen aus. Für 250 Anmeldungen mit PIM-SM ergeben sich jeweils berechnet/simuliert für TRS-R 2349/1628, TRS-K 544/576, TRS-M 3004/930 und ERS 95125/5830 Nachrichten. Nicht verwunderlich ist die geringere gemessene Nachrichtenzahl von TRS-R und vor allem ERS, da keine Hintergrundlast beaufschlagt war. In Simulationen mit beaufschlagter Hintergrundlast von 19% und einer Wartezeiteinstellung von 0,1 Sekunden wurde von ERS bereits mit DVMRP ungefähr 50% der in Kapitel 3.9.1 berechneten

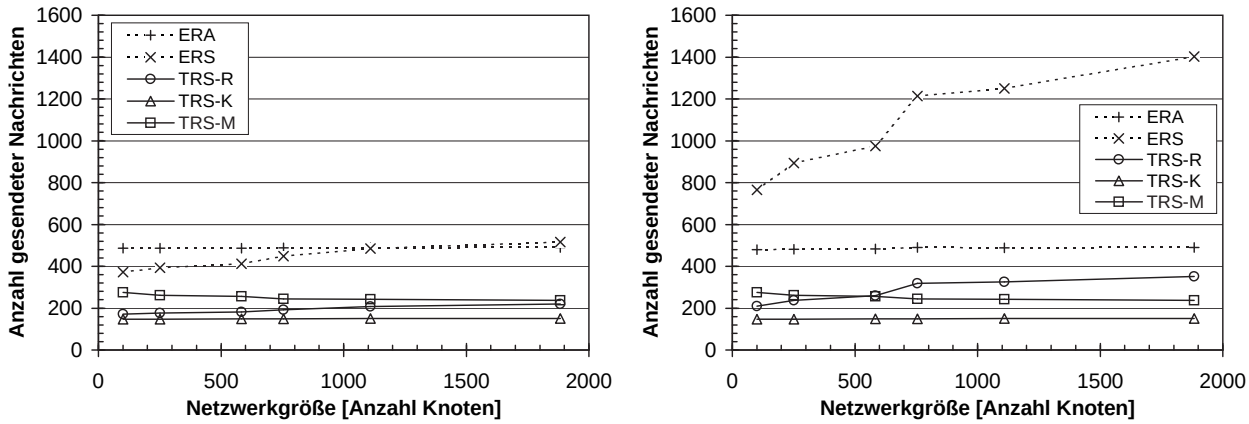



*Abbildung 3.33: Gesendete Nachrichten in Abhängigkeit von der Netzwerkgröße a) DVMRP– und b) PIM-SM-Routing*

maximalen Nachrichtenzahl erreicht, die mit PIM-SM nochmals erhöht worden wäre. TRS-M bleibt ebenfalls deutlich unter den berechneten Maximalwerten, da mit der Kontrollbaumhöhe eine Metrik verwendet wurde, die nur in wenigen Fällen globale Markensuchen erfordert. Ungewöhnlich hingegeben ist die gemessene Nachrichtenanzahl für TRS-K, die den berechneten Maximalwert überschreitet. Dies ist kein Fehler in den Berechnungen oder in der Simulation, sondern eine Abweichung der (simulierten) Realität vom Modell der Berechnung. Die Berechnung basierte explizit auf der Annahme, dass neue Teilnehmer streng sequentiell angemeldet werden. Das heißt, eine Anmeldung und die sich anschließende Aktualisierung der Gruppenregister, wird vollständig ausgeführt, bevor die nächste Anmeldung erfolgt. Dies ist in der Simulation und auch in der Realität nicht gegeben. Die hier ablaufenden nebenläufigen Markensuchen und Gruppenregisteraktualisierungen führen in Einzelfällen dazu, dass für eine Markensuche ein größerer Bereich der TRS-Hierarchie abgesucht wird als im sequentiellen Fall, was schließlich die zusätzlich gemessenen Nachrichten verursacht.

#### 3.10.2.3 Nachrichtenaufwand in Abhängigkeit von der Netzwerkgröße

Abbildung 3.33 illustriert die Gesamtanzahl gesendeter Nachrichten für den Kontrollbaumaufbau in Abhängigkeit von der Netzwerkgröße, d.h. der Anzahl Knoten im Netzwerk, für 50 anzumeldende Mitglieder. Wie soeben beschrieben, ist die Anzahl empfangener Nachrichten für TRS-K und TRS-M identisch zu den gesendeten Nachrichten, während sich für TRS-R und ERS moderate Abweichungen ergeben. Für ERA ist die Anzahl empfangener Nachrichten ca. zehnmal höher als die Anzahl gesendeter Nachrichten, was für eine Beurteilung der tatsächlichen Netzwerklast und Knotenlast berücksichtigt werden muss.

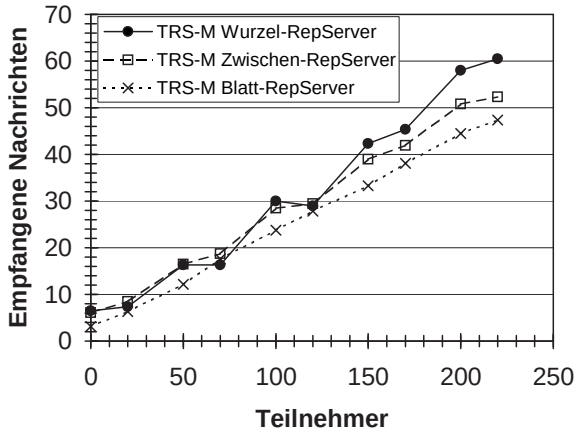
In Teilabbildung 3.33a findet das Routing-Protokoll DVMRP Verwendung. Wiederum resultiert der TRS-Dienst in der niedrigsten Nachrichtenanzahl. Während die Ergebnisse für TRS-M und TRS-K bis

auf zufällige Schwankungen konstant sind, zeigt sich für TRS-R eine leichte Zunahme der Nachrichtenanzahl mit der Netzwerkgröße. Der größere mittlere Netzwerkdurchmesser bei konstanter Anzahl TRS-RepServer erfordert mehr zu sendende Suchnachrichten, bis ein RepServer mit Marken gefunden werden kann. Deutlicher wird dieser Effekt bei ERS sichtbar. Nicht nur die Zunahme der Nachrichtenanzahl fällt drastischer aus, auch absolut erfordert ERS deutlich mehr Nachrichten als die TRS-Varianten. ERA ist wiederum unabhängig von der Netzwerkgröße, verursacht aber dennoch einen höheren Aufwand als die TRS-Varianten. Für ausreichend große Netze wäre es denkbar, dass ERA einen kleineren Aufwand verursacht als das nicht konstante TRS-R. In der Praxis ist dieser Fall kaum wahrscheinlich, da der Nachrichtenaufwand bei ERA von der Lebenszeit einer Multicast-Gruppe abhängt. Dies wiegt wesentlich schwerer als die geringe Abhängigkeit von der Netzwerkgröße, die bei TRS-R im Gegensatz zu ERS zudem nur solange gegeben ist, bis alle TRS-RepServer Marken besitzen. Daraus lässt sich schließen, dass eine größere Teilnehmerzahl zu einer wesentlich geringeren Abhängigkeit von der Netzwerkgröße bei TRS-R führen wird. Für ERS trifft dies nicht zu. Zu beachten ist zudem, dass eine gesendete Multicast-Nachricht von ERS, ERA und TRS-R ebenfalls lediglich als eine Nachricht, analog zu einer Unicast-Nachricht, gewichtet ist. Das heißt, die tatsächliche Netzwerkbelastung dieser Verfahren, vor allem von ERS und ERA — diese verwenden hauptsächlich Multicast — ist höher als die Abbildung widerspiegelt. Dies unterstreicht die Überlegenheit der Verfahren basierend auf einer RepServer-Hierarchie, TRS-K und TRS-M.

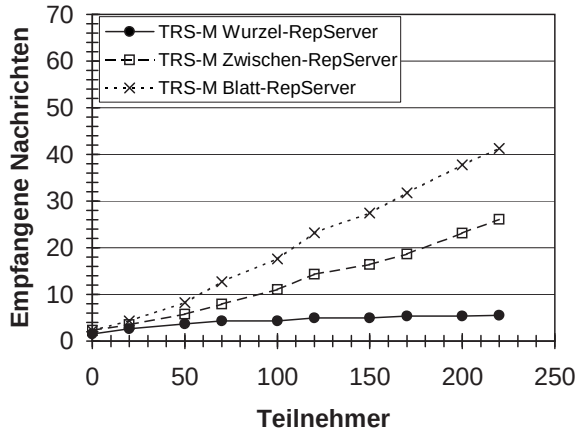
Bei der Betrachtung von Abbildung 3.33b fällt das schlechte Verhalten von ERS mit PIM-SM-Routing ins Auge. Das dabei erfolgende Aussenden aller Nachrichten über eine gemeinsame Rendezvous-Stelle im Netzwerk führt zu einer großen Anzahl notwendiger Nachrichten. Dieses Verhalten wurde aufgrund der Überlegungen von Abschnitt 3.2.3 noch nicht vorhergesagt, lässt sich aber dennoch einfach erklären. Die Ausbreitung aller Suchnachrichten von immer derselben Stelle im Netzwerk führt dazu, dass die Kontrollbaummitglieder in deren Nähe schnell ihren maximalen Verzweigungsgrad erreichen. So muss ein immer größerer Bereich abgesucht werden, anstatt ein immer kleinerer, wie es bei DVMRP-Routing mit zunehmender Teilnehmerzahl der Fall ist. Auch TRS-R zeigt ein verschlechtertes Verhalten mit PIM-SM. ERA, TRS-K und TRS-M sind bezüglich der Nachrichtenanzahl unabhängig vom Multicast-Routing-Protokoll.

#### 3.10.2.4 Nachrichtenaufwand der RepServer

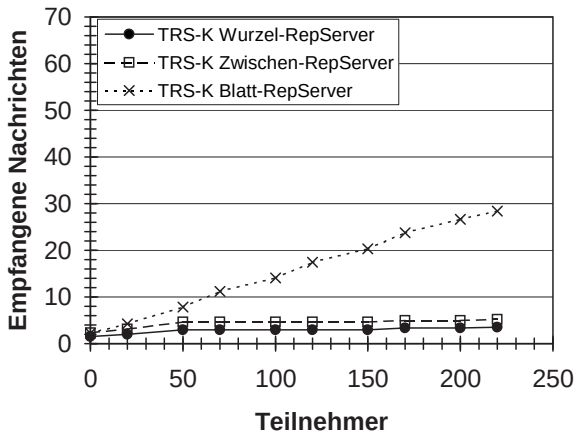
Trotz der gezeigten guten Skalierbarkeit von TRS für das Netzwerk bleibt die Unsicherheit, ob nicht innerhalb der RepServer-Hierarchie ein Flaschenhals auftritt. Im Gegensatz zu der bisher untersuchten Gesamtanzahl gesendeter oder empfangener Nachrichten von RepServern *und* Multicast-Teilnehmern, was die Last im Netzwerk widerspiegelt, wird in diesem Abschnitt nur die Anzahl empfangener Nachrichten eines einzelnen RepServers betrachtet, was die Last eines RepServers widerspiegelt. Ein Flaschenhals könnte beispielsweise an der Wurzel auftreten, d.h. ein RepServer könnte durch zu viele




(a) TRS-M RepServer ohne Toleranz




(b) TRS-M RepServer mit Toleranz zwei




(c) TRS-K RepServer

*Abbildung 3.34: Belastung eines einzelnen TRS-RepServers mit empfangenen Nachrichten*

zu verarbeitende Nachrichten überlastet werden. In Abschnitt 3.8.2 wurde das Problem bereits analytisch untersucht mit dem Ergebnis, dass eine Überlastung unwahrscheinlich ist. Zudem wurde für den Fall einer Überlastung ein einfacher Ausweg aufgezeigt. Die folgenden Simulationsergebnisse sollen die Analyse bestätigen.

In Abbildung 3.34 ist die Belastung eines einzelnen Blatt-, Zwischen- oder Wurzel-RepServer bezüglich der empfangenen Nachrichten in Abhängigkeit von der Multicast-Gruppengröße dargestellt. Teilabbildung a) zeigt das Ergebnis für TRS-M ohne Toleranz, während in Teilabbildung b) TRS-M mit einer Toleranz von zwei dargestellt ist, d.h. mit einer erlaubten Abweichung der Höhe des entstehenden Kontrollbaums um zwei Hierarchiestufen vom Minimalwert. Schließlich zeigt Teilabbildung c) die empfangenen Nachrichten eines RepServers mit dem Protokoll TRS-K.

Zu beachten ist, dass die Ergebnisse für steigende Teilnehmerzahlen nicht einer einzelnen Simula-

tion mit Zwischenergebnissen entstammen, sondern für jede Teilnehmerzahl eine eigene Simulation mit zufälliger zeitlicher und räumlicher Verteilung der Teilnehmer durchgeführt wurde. Daraus erklärt sich, dass sich die Nachrichtenanzahl, bedingt durch zufällige Schwankungen der zeitlichen und räumlichen Verteilung von Anmeldungen, bei erhöhter Teilnehmerzahl auch verringern kann.

Die Teilabbildung a) für TRS-M zeigt einen Anstieg der Nachrichtenbelastung aller drei RepServer-Typen mit steigender Teilnehmerzahl. Alle drei RepServer-Typen zeigen allerdings einen Anstieg, der bei angenommener Linearität eine kleinere Steigung als eins aufweist. Für die Such-RepServer außer der Wurzel und die Blatt-RepServer ergibt sich dieses Ergebnis aus der Anzahl von sechs Zwischen-RepServern und acht Blatt-RepServern, die für eine entsprechende Lastverteilung sorgen. Da dies für den Wurzel-RepServer nicht zutrifft, ist dessen Nachrichtenzunahme echt kleiner als linear mit Steigung eins. Die analytischen Überlegungen aus Abschnitt 3.8.2 resultierten lediglich in einer logarithmischen Nachrichtenzunahme für den RepServer. Dies kann abschließend mit der vorliegenden Simulation nicht nachgewiesen werden. Dazu ist der simulierte Bereich mit 220 Anmeldungen zu gering, um eindeutig zwischen linear und logarithmisch unterscheiden zu können. Allerdings ist dies aufgrund der Messungen zumindest plausibel.

Im Vergleich mit der Teilabbildung b) zeigt sich, dass sich bereits mit einer Toleranz von zwei gegenüber vollständig höhenbalancierten Kontrollbäumen eine deutliche Entlastung, vor allem des Wurzel-RepServers, aber auch der Zwischen-RepServer (Such-RepServer ohne den Wurzel-RepServer), erreichen lässt. Die Nachrichtenbelastung des Wurzel-RepServers wächst hier nur noch äußerst langsam mit der Teilnehmerzahl. Erwartungskonform ergibt sich kaum eine Entlastung der Blatt-RepServer, da diese weiterhin alle Teilnehmer mit einer Marke versorgen müssen. Lediglich eine kleine Reduzierung der Markenlieferungen an andere Blatt-RepServer tritt ein. Dieses Verhalten beeinflusst die Skalierbarkeit nicht nachteilig, da durch eine Erhöhung der Anzahl an Blatt-RepServern eine Entlastung derer einfach erreicht werden kann.

Schließlich zeigt Teil c) der Abbildung 3.34 die Ergebnisse für TRS-K. Die Nachrichtenbelastung ist gegenüber TRS-M mit Toleranz zwei nochmals gesunken, für Zwischen-RepServer gar sehr deutlich. Für alle Such-RepServer zeigt sich nach einer anfänglichen Zunahme kaum mehr eine weitere Zunahme der Nachrichtenanzahl, da alle Blatt-RepServer eine Marke besitzen und keine weiteren Markensuchen mehr erfolgen.

Kontrollmessungen zu TRS-M mit TRS-Bäumen anderer Verzweigungsgrade und Höhen ergaben, dass eine Erhöhung der Anzahl an RepServern ebenfalls zu einer Erhöhung des globalen Nachrichtenaufwands führt. Dies erklärt sich aufgrund der zunehmenden Anzahl an Markensuchen und abzufragender RepServer im Verlauf einer Markensuche. Beispielsweise nahm mit der Erhöhung der Anzahl an RepServern von 15 auf 85 eines simulierten Szenarios die Nachrichtenanzahl von 380 Nachrichten auf 784 Nachrichten zu. Allerdings geht damit eine antiproportionale Entlastung der Blatt-RepServer von durchschnittlich 22 auf 5 Nachrichten pro Blatt-RepServer einher. Die Last der

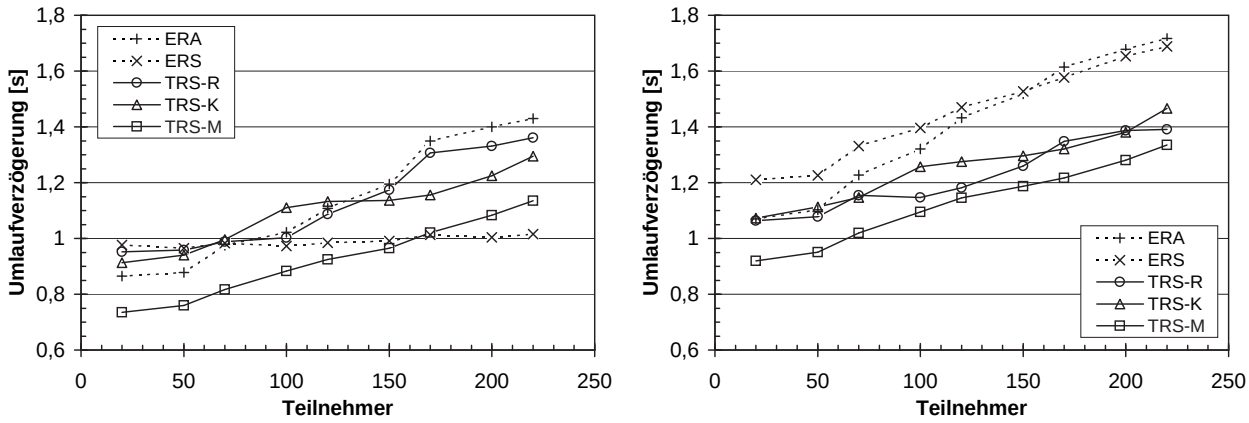



*Abbildung 3.35: Mittlere Umlaufverzögerung im Kontrollbaum a) DVMRP und b) PIM-SM Routing*

Wurzel lies sich als proportional zum Verzweigungsgrad des TRS-Baums ermitteln, da deren Last für Markensuchen und Aktualisierungen der Gruppenregister direkt von der Anzahl Kindknoten abhängt. Für die Zwischen-RepServer konnten keine signifikanten Zusammenhänge festgestellt werden.

### 3.10.2.5 Verzögerung des Kontrollbaums

Nicht nur die Skalierbarkeit soll zur Bewertung der verschiedenen Verfahren herangezogen werden, sondern auch die Eigenschaften der entstehenden Kontrollbäume. Die Verzögerung innerhalb des Kontrollbaums ist dabei ein wichtiges Kriterium. In Abbildung 3.35 ist die durchschnittliche akkumulierte Umlaufverzögerung aller Verbindungen zwischen den Multicast-Teilnehmern im Kontrollbaum und des Senders dargestellt. Obwohl diese Verzögerung die initiale Datenübertragung nicht beeinflusst — diese erfolgt schließlich ohne Mithilfe des Kontrollbaums — wird die Zeit für Übertragungswiederholungen dadurch bestimmt.

Für DVMRP-Routing liegen die Verzögerungszeiten eng zusammen. Ein Vorteil lässt sich dennoch für ERS und TRS-M gegenüber den anderen Verfahren erkennen. Hierzu sei angemerkt, dass das gute Abschneiden von TRS-M und die Ergebnisse von TRS-R und TRS-K einen großen Erfolg darstellen. Schließlich besteht das Prinzip von ERS darin immer den nächstliegenden Knoten, bezüglich der Anzahl von Teilstrecken und als zweites Kriterium der Verzögerung zu ermitteln und ist folglich für die ermittelte Größe bestens geeignet. Wie sich beim Vergleich mit Teilabbildung b) zeigt, gelingt dies allerdings einzig mit DVMRP-Routing. Wird PIM-SM-Routing eingesetzt, so zeigen sich deutliche Nachteile für ERS und ERA, die mit rendezvousbasiertem Routing nicht mehr in der Lage sind, die Verzögerungen zu optimieren. Dies muss unmittelbar verständlich der Fall sein, da sich bei PIM-SM die Multicast-Pfade deutlich von den Unicast-Pfaden unterscheiden, deren Verzögerung hier gemessen wurde. Selbst der Einsatz von Multicast für das Senden von Quittungen und Übertragungswiederholungen würde diese Situation nicht verbessern, da deren Auslieferung über die Rendezvous-Stelle

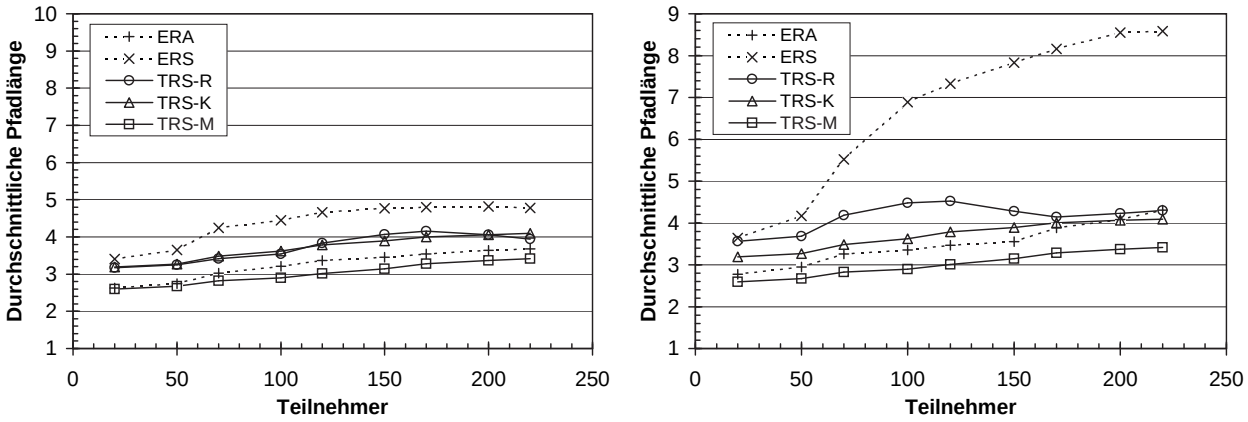



*Abbildung 3.36: Mittlere Pfadlänge des Kontrollbaums und die daraus resultierende Zuverlässigkeit a) DVMRP und b) PIM-SM Routing*

bei PIM-SM ebenfalls kontraproduktiv wäre — lokale Gruppen könnten aufgrund dieses Routing-Baums nicht etabliert werden — und deswegen nur Unicast Verwendung finden kann. Bemerkenswert gut ist unter dieser Betrachtung das Abschneiden von TRS-R. Die Auswahl von Vaterknoten der gleichen Domäne bei TRS-R ist erfolgreicher als der Versuch, nahe liegende Vaterknoten über eine Multicast-Suche zu finden. Insgesamt kann wiederum TRS-M die geringste Nachrichtenverzögerung bescheinigt werden.

#### 3.10.2.6 Höhe des Kontrollbaums

In Abbildung 3.36 ist als zweite wichtige Eigenschaft des Kontrollbaums dessen mittlere Pfadlänge aufgetragen. Der Zusammenhang zwischen der Pfadlänge im Kontrollbaum und der Zuverlässigkeit wurde bereits in Abschnitt 3.5.2 diskutiert. Das optimale Ergebnis erreicht zwangsläufig TRS-M. Dessen mittlere Pfadlänge entspricht der berechenbaren mittleren Pfadlänge von balancierten Bäumen und bestätigt die Analyse aus Abschnitt 3.9. Ungewöhnlich gute Resultate, die deutlich besser liegen als die Maximalwerte der Analyse, erzielt auch ERA. Gegenüber ERS wird bei ERA keine perfekte Optimierung nach der kürzesten Netzwerkdistanz zwischen Vater– und Kindknoten durchgeführt, da entsprechend der hier gewählten Konfiguration ein Knoten dazu nicht lange genug auf eintreffende Suchnachrichten wartet respektive die Suchnachrichtenrate zu gering ist. Bereits in Abbildung 3.35 wurde dies durch höhere Verzögerungen von ERA mit DVMRP-Routing ersichtlich. Auch weiter entfernt liegende Knoten haben gelegentlich die Chance, als Vaterknoten gewählt zu werden. Vor allem die zeitlich ersten Knoten im Kontrollbaum können mit hoher Wahrscheinlichkeit viele Kindknoten gewinnen, was schließlich die gewünschte Auffächerung des Kontrollbaums ermöglicht. Durch die Analyse wurde dies nicht berücksichtigt und eine Berücksichtigung war auch nicht erwünscht, da die analysierte *maximale* Kontrollbaumhöhe weiterhin ihre Berechtigung besitzt. Bereits eine Änderung

der Parameter, beispielsweise eine erhöhte Senderate der Suchnachrichten oder erhöhte Wartezeit der neuen Teilnehmer auf Suchnachrichten naher Knoten, würde das hier interessierende Verhalten von ERA verschlechtern und dem von ERS annähern.

Während sich TRS-R und TRS-K im Mittelfeld befinden, sind die mittleren Pfadlängen von ERS ungefähr 50% höher als das Optimum, was die geringste Zuverlässigkeit des Kontrollbaums bedeutet. Wird PIM-SM als Routing-Verfahren eingesetzt, so verschlechtern sich die Resultate für ERS nochmals erheblich und steigen zudem beträchtlich mit der Teilnehmerzahl. Auch für ERA und TRS-R ergeben sich größere Pfadlängen, allerdings fallen die Unterschiede zu DVMRP moderat aus. TRS-K und TRS-M liefern die exakt identischen Resultate, da sie kein Multicast verwenden und deshalb keine Abhängigkeit vom Multicast-Routing-Protokoll aufweisen. Im Vergleich mit den analysierten *maximalen* Kontrollbaumhöhen aus Abschnitt 3.9 bleiben alle Verfahren, bis auf TRS-M, deutlich unter den Berechnungen. Dennoch zeigt sich das aus der Analyse prognostizierte schlechte Verhalten von ERS. Bei einer Änderung der Wartezeiteinstellungen für ERA wie oben ausgeführt, zeigt auch ERA ähnlich schlechte Ergebnisse wie ERS, was durch Kontrollmessungen bestätigt werden konnte.

Die Ergebnisse aller Simulationen werden in Abschnitt 3.12 nochmals zusammengefasst.

## 3.11 Eingliederung in Transportprotokolle

Multicast-Transportprotokolle wurden oftmals unter Verwendung von ERS oder ERA zum Kontrollbaumaufbau [Levine & Garcia-Luna-Aceves 1998, Yavatkar et al. 1995, Chiu et al. 1998] oder sogar gänzlich ohne vollautomatische Kontrollbaumkonfiguration spezifiziert [Paul et al. 1997]. Transportprotokolle lassen sich ebenfalls einfach mit dem TRS-Dienst kombinieren. Beispiele dafür sind RMTP [Paul et al. 1997] und TMTP [Yavatkar et al. 1995], die in einer Simulationsumgebung zusammen mit TRS implementiert wurden und für die Simulationsergebnisse in Abschnitt 4.7 beschrieben werden. Dazu wurden deren eigene Mechanismen durch Aufrufe an den TRS-Dienst ersetzt. Anstatt die proprietären Schnittstellen dieser Protokolle zu beschreiben, wird hier eine allgemeine Schnittstelle beschrieben, die in [Kadansky et al. 2001] diskutiert wird. Der Internet-Draft zielt darauf ab, allgemeine Konzepte und eine Schnittstelle für den Kontrollbaumaufbau zu beschreiben, die mit verschiedenen Dienstmodellen, Routing-Protokollen und Transportprotokollen sowie mit verschiedenen Verfahren zum Kontrollbaumaufbau kompatibel sind. Der Internet-Draft und dessen aktuelles Datum belegen zudem, dass die Notwendigkeit eines skalierbaren, großflächig verwendbaren und flexiblen Kontrollbaumaufbaus nun auch von anderen erkannt wurde.

Für den Kontrollbaumaufbau werden in [Kadansky et al. 2001] vier Lösungen definiert: (1) Statisch, (2) ERS, (3) Mesh und (4) Point-of-Contact (POC). Der statische Ansatz erlaubt keinen automatischen Kontrollbaumaufbau und ERS wurde bereits ausführlich erläutert. Mesh ist ein spezialisierter Ansatz,

in dem die Knoten des Kontrollbaums untereinander Informationen austauschen, der aber das Problem einen Knoten des Kontrollbaums zu finden nicht selbst löst, sondern auf einen der anderen Ansätze angewiesen ist. Im Rahmen dieser Arbeit ist der POC-Ansatz von Interesse.

Der POC-Ansatz arbeitet folgendermaßen. Vorausgesetzt wird die Existenz von bestimmten Knoten, genannt POCs. Diese sind im Allgemeinen nicht Mitglied eines Kontrollbaums, haben aber Kenntnisse über die Kontrollbäume und die freien Plätze für Kindknoten. Möchte sich ein neues Mitglied an den Kontrollbaum anmelden, wird einer dieser Knoten nach einem Gruppenführer im Kontrollbaum gefragt. Die POCs selbst können laut der Beschreibung entweder ein wohlbekannter Dienst sein oder durch einen geeigneten Mechanismus wie ERS oder Anycast [Partridge et al. 1993] bestimmt werden. Wie ein POC intern den Dienst realisiert, ist nicht Inhalt der zitierten Beschreibung und soll auch bewusst nicht eingeschränkt werden. Definiert ist zudem, dass die Mitglieder des Kontrollbaums, die weitere Kindknoten akzeptieren wollen, periodisch diese Information an die POCs weiterreichen. Dies verträgt sich sehr gut mit dem hier vorgestellten Soft-State-Ansatz für die Aktualisierung der Markeninformation.

Die Schnittstelle zwischen dem Transportprotokoll und dem POC-Dienst ist nicht identisch zu der in Abschnitt 3.4 vorgestellten Schnittstelle. Dennoch lassen sich die notwendigen Funktionalitäten zur Realisierung des TRS-Diensts finden. In Tabelle 3.6 sind die entsprechenden äquivalenten Operationen der Schnittstellen aufgelistet. Auf eine Angabe der Übergabeparameter wurde verzichtet, da diese in besagtem Internet-Draft noch nicht vollständig sind und zudem vielfach zusätzliche optionale Parameter erlauben.

Wie aus der Tabelle ersichtlich wird, finden sich alle Operationen des TRS-Diensts auch in der POC-Schnittstelle wieder oder lassen sich nachbilden. Die Operation zum Löschen einer Gruppe muss aus einzelnen *removeChild*-Operationen nachgebildet werden. Die *repJoinGroup*-Operation wird auf zwei Operationen der POC-Schnittstelle abgebildet, aufgrund dessen, dass die Abfrage möglicher Vaterknoten im Kontrollbaum und die Auswahl eines Vaterknotens getrennte Vorgänge sind. Insgesamt zeigt sich eine große Ähnlichkeit und folglich die einfache Möglichkeit, die POC-Schnittstelle zwischen TRS-Dienst und Transportprotokoll zu nutzen, sobald diese breite Verwendung findet.

## 3.12 Diskussion

Expanding ring search (ERS) und expanding ring advertisement (ERA) sind auf den ersten Blick völlig ausreichende Protokolle zum Aufbau von Kontrollbäumen. Beide benötigen keinerlei Unterstützung durch eine Server-Infrastruktur und sind einfach zu realisieren. Ein weiterer Vorteil ist ihre inhärente Fehlertoleranz.

| TRS-Operation | POC-Operation | Beschreibung |
|---|---|---|
| repCreateGroup | acceptChild | Mit acceptChild wird der POC-Dienst angewiesen, einen neuen Knoten als Mitglied zu akzeptieren. |
| repDeleteGroup | removeChild | Es ist keine Operation vorgesehen, um eine komplette Gruppe zu löschen. Daher muss removeChild rekursiv angewendet werden. |
| repJoinGroup | getSNs und setSN | Mit getSNs (SN = Service Node, hier Gruppenführer im Kontrollbaum) werden eine oder mehrere Gruppenführer im Kontrollbaum erfragt. Nachdem die Anmeldung stattgefunden hat, kann dies mit setSN dem POC-Dienst mitgeteilt werden. |
| repLeaveGroup | removeChild | Siehe repDeleteGroup. |
| repAddToken | acceptChild | Siehe repCreateGroup. |
| repRefreshToken | advertise | Mit advertise wird ein Mitglied des Kontrollbaums dem POC-Dienst bekannt gemacht und die Information aufgefrischt. |

*Tabelle 3.6: Gegenüberstellung der Schnittstelle zwischen dem TRS-Dienst und dem POC-Dienst*

Allerdings wurden die Skalierbarkeit und die Effizienz von ERS und ERA zuvor noch nie einer ausführlichen Evaluation unterzogen. Eine solche Untersuchung muss vor allem große Gruppen berücksichtigen, da hierarchische Protokolle ja gerade bei großen Gruppen eingesetzt werden und dann deutliche Vorteile gegenüber flachen Ansätzen besitzen. Doch für große Gruppen erweisen sich ERS und ERA, aufgrund des analysierten maximalen Nachrichtenaufwands, als ungeeignet, da sie nicht skalierbar sind. Damit ist der vorgestellte TRS-Dienst derzeit der einzige skalierbare Ansatz zum Aufbau von Kontrollbäumen auf der Transportschicht.

ERS und ERA besitzen noch weitere Nachteile, die ihren Einsatz ebenso fragwürdig erscheinen lassen. Besonders ihre universelle Einsetzbarkeit ist durch zahlreiche Vorbedingungen stark eingeschränkt. So benötigen sie einen Multicast-Dienst auf Vermittlungsschicht und ERS benötigt sogar einen bidirektionalen Multicast-Dienst, was in Satellitennetzwerken beispielsweise nicht gegeben ist. Bezüglich des Routing-Protokolls gibt es kein ideales Verfahren für ERS und ERA. Mit rendezvousbasierten Verfahren können keine geeigneten Kontrollbäume aufgebaut werden und die Verkehrskonzentration an der Rendezvous-Stelle ist ebenso nachteilig. Auf der anderen Seite wird mit senderbasierten Verfahren ein zu hoher Aufwand auf der Vermittlungsschicht durch den Aufbau der Spannbäume erzeugt. Mit senderspezifischem PIM, das gute Aussichten besitzt, das zukünftige Standardprotokoll für Multicast-Routing zu werden, arbeiten ERS und ERA erst gar nicht zusammen. Damit ist TRS nicht nur der einzige skalierbare Dienst, sondern je nachdem wie die Entwicklung voranschreitet vermutlich sogar der einzige Dienst, der flächendeckend eingesetzt werden kann.

Der TRS-Dienst ist erstmal lediglich ein Konzept bzw. eine Dienstspezifikation, die beschreibt, wie Kontrollbäume durch die Abfrage von Marken aufgebaut werden können. Die interne Realisierung kann auf verschiedene Arten erfolgen. Eine zentralisierte Realisierung wurde nicht vorgestellt, da die Skalierbarkeit und Fehlertoleranz ungenügend ausgeprägt wären. Vorgestellt wurden dagegen drei verteilte Strategien, die unterschiedliche Charakteristiken besitzen. TRS-R kann einfach realisiert werden, behebt indessen nicht alle der oben zusammengefassten Probleme von ERS und ERA, da intern auf eine erweiternde Ringsuche zurückgegriffen werden muss. Mit TRS-K und TRS-M werden die Nachteile von ERS und ERA endgültig überwunden. Dabei kann TRS-M Kontrollbäume durch den Einsatz von Metriken global optimieren, besitzt dadurch aber auch einen höheren internen Kommunikationsaufwand. Mit deren Hilfe kann die Zuverlässigkeit des entstehenden Kontrollbaums maximiert werden oder der Kontrollbaum für den häufigen Spezialfall eines Senders optimiert werden. Im weiteren Verlauf der Arbeit wurde dargelegt, wie der TRS-Dienst fehlertolerant und skalierbar implementiert werden kann. Schließlich wurde ausgeführt, wie der TRS-Dienst in zuverlässige Transportprotokolle eingegliedert werden kann.

Die quantitativen Vorteile von TRS gegenüber ERS und ERA wurden durch Analyseergebnisse und Simulationsergebnisse dargelegt. Dabei hat sich gezeigt, dass der TRS-Dienst durchweg besser abschneidet. Bezüglich der erzielten Verzögerungen im Kontrollbaum war das Ergebnis knapp, da ERS mit DVMRP sehr gute Resultate erzielt und einzig von TRS-M in einigen Messungen unterboten wird. Im Gegensatz dazu schnitten alle TRS-Varianten mit PIM-SM Routing besser als ERS und ERA ab. Für die Zuverlässigkeit des entstehenden Kontrollbaums bietet TRS-M ebenfalls die besten Ergebnisse. Am deutlichsten wird die Überlegenheit von TRS bei allen Messungen bezüglich der Skalierbarkeit. Hier resultieren ERS und ERA oftmals in einem derart hohen Nachrichtenaufwand, dass deren weiträumiger Einsatz zumindest fragwürdig wenn nicht unmöglich erscheint. Nur der skalierbare TRS-Dienst, alle Realisierungsvarianten eingeschlossen, ist aufgrund der dargelegten Messungen in der Lage, bei umfangreichem Einsatz eines zuverlässigen Multicast-Diensts mit dem Kontrollbaumaufbau betraut zu werden.

# Kapitel 4

# Analyse zuverlässiger Multicast-Transportprotokolle

## 4.1 Einführung

Durch die Verwendung von Kontrollbäumen und der damit einhergehenden Lastverteilung auf viele Gruppenführer, versprechen hierarchische Multicast-Tranportprotokolle Skalierbarkeit auch für große Teilnehmerzahlen. Mit dem Token-Repository-Service wurde erstmals ein effizienter Dienst zum Aufbau von Kontrollbäumen für hierarchische Transportprotokolle vorgestellt. Im weiteren Verlauf dieses Kapitel werden die bisher nur behaupteten Vorteile hierarchischer Transportprotokolle gegenüber flachen Transportprotokollen detailliert mit Analysen und Simulationen belegt und damit die Notwendigkeit für den TRS-Dienst untermauert.

Der TRS-Dienst ermöglicht aber im Vergleich zu alternativen Ansätzen nicht nur die skalierbare Erstellung von Kontrollbäumen. Wie im vorigen Kapitel und besonders in Abschnitt 3.7 ausgeführt wurde, erlaubt der TRS-Dienst zudem die Eigenschaften der entstehenden Kontrollbäume flexibel zu beeinflussen. Die wichtige Eigenschaft des Verzweigungsgrads der entstehenden Kontrollbäume wird im Verlauf dieses Kapitels ausführlich betrachtet. Dabei zeigt sich, dass in vielen Fällen ein mit dem TRS-Dienst optimierter Verzweigungsgrad die Last im Netzwerk verringern sowie den Durchsatz und die Verzögerung eines hierarchischen Transportprotokolls nochmals signifikant verbessern kann. Die dargelegten Ergebnisse können Verbesserungen im Bereich bis zu einer Größenordnung nachweisen.

Im Folgenden werden dazu sieben repräsentative Protokollklassen untersucht und miteinander verglichen. Diese teilen sich auf in drei flache und vier hierarchische Protokollklassen. Die betrachteten Protokollklassen beinhalten die geläufigsten Techniken zur Realisierung von zuverlässigem Multicast. Eine ausführliche Beschreibung der Protokollklassen wird im nächsten Abschnitt gegeben. Danach

erfolgt eine ausführliche Analyse der verursachten CPU-Belastung, des verursachten Bandbreitenbedarfs und der resultierenden Nachrichtenverzögerung beim Einsatz dieser Protokollklassen. Die Analyse schließt mit numerischen Ergebnissen zu den interessantesten Gesichtspunkten ab, beispielsweise Ergebnisse zum Durchsatz, zur Verzögerung und zum Einfluss des Verzweigungsgrads. In einem weiteren Abschnitt werden die numerischen Ergebnisse der Analyse mit Simulationsergebnissen verglichen, um die Stichhaltigkeit der Analyse zu bekräftigen. Schließlich werden in der Diskussion die wichtigsten Ergebnisse zusammengefasst und Konsequenzen für den Token-Repository-Service abgeleitet.

## 4.2 Klassifikation der untersuchten Protokolle

Eine Klassifikation der zuverlässigen Multicast-Protokolle kann nach mehreren Gesichtspunkten erfolgen. Die geläufigste Einteilung ist nach der Art der Quittungsmeldungen und ob eine hierarchische Kontrollstruktur für lokale Übertragungswiederholungen Verwendung findet. In Kapitel 2.5.3 wurde bereits eine Klassifikation gegeben. Abbildung 4.1 fasst die hier verwendete und erweiterte Klassifikation zusammen. Ähnliche, aber nicht so weitreichende Klassifikationen lassen sich auch in [Pingali et al. 1994], [Levine & Garcia-Luna-Aceves 1998] und [Nonnenmacher et al. 1998] finden. Im Verlauf dieses Kapitels wird von einem einzigen Sender pro Multicast-Gruppe ausgegangen.

### 4.2.1 Senderinitiiertes Protokoll (K-ACK)

Die Tatsache, dass der Sender für die Erkennung von Datenverlust zuständig ist, hat dieser Protokollklasse den Namen verliehen. Die Klasse der *senderinitiierten Protokolle* verwendet positive Quittungen (ACKs), die von den Empfängern an den Sender mittels Unicast übertragen werden, nachdem die Datennachricht erfolgreich empfangen wurde. Eine fehlende Quittung signalisiert dem Sender, dass entweder die Datennachricht oder aber die Quittung verloren ging oder der Empfänger ausgefallen ist. Da dies für den Sender nicht unterscheidbar ist, führt eine verlorene Quittung generell zu einer Übertragungswiederholung. Obwohl es diese Klasse erlauben würde, einzelnen Empfängern selektiv eine Übertragungswiederholung zukommen zu lassen, wird aufgrund der Vergleichbarkeit mit früheren Arbeiten [Pingali et al. 1994] angenommen, dass eine Übertragungswiederholung prinzipiell mittels eines Multicast-Diensts an die gesamte Gruppe gesendet wird.

Da viele reale Protokolle sowohl positive als auch negative Quittungen verwenden, muss die Definition noch präzisiert werden, um eine eindeutige Klassifikation zu erlauben. Protokollen aus dieser Klasse ist es durchaus erlaubt, auch negative Quittungen zu verwenden. Dies kann beispielsweise sinnvoll sein, um eine schnelle Übertragungswiederholung anzustoßen. Jedoch ist gefordert, dass in

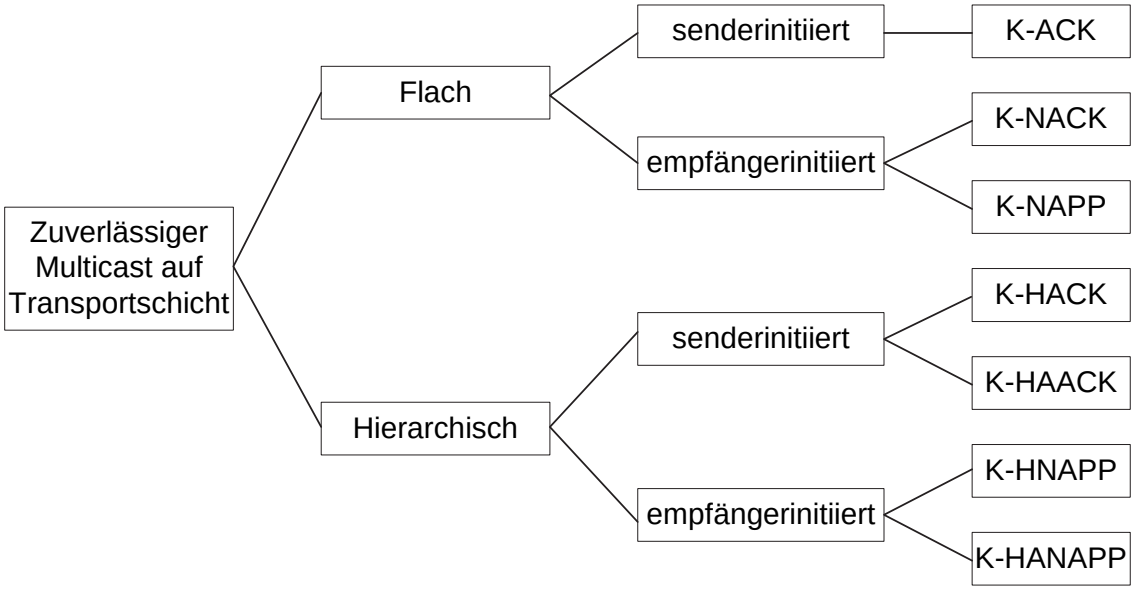



*Abbildung 4.1: Klassifikation zuverlässiger Multicast-Transportprotokolle*

jedem Falle auch positive Quittungen notwendig sind, z.B. um die Daten aus dem Sendepuffer zu entfernen.

Das in Abschnitt 2.5.3 besprochene XTP-Protokoll [Strayer et al. 1992] ist ein Beispiel für diese Protokollklasse. Als Bezeichnung der Protokollklasse wird K-ACK verwendet. „K“ kennzeichnet, dass es sich um eine Klasse handelt und „ACK“ kennzeichnet die notwendigen positiven Quittungen. Die Kennzeichnung dieser Klasse als senderinitiiert ist nicht hinreichend, angesichts hierarchischer senderinitiierter Protokolle, die von der vorliegenden Protokollklasse unterschieden werden sollen.

### 4.2.2 Empfängerinitiiertes Protokoll (K-NACK)

Im Gegensatz zu den senderinitiierten Protokollen ist es den *empfängerinitiierten Protokollen* nicht gestattet positive Quittungen zu versenden. Sobald ein Empfänger einen Fehler entdeckt, wird eine negative Quittung (NACK) mittels Unicast an den Sender übermittelt. Ein Fehler wird entdeckt durch einen Sprung in den Sequenznummern, durch eine falsche Kontrollsumme oder durch eine Zeitüberschreitung beim Warten auf die nächste Nachricht. Analog zu der Klasse K-ACK wird diese Protokollklasse als K-NACK gekennzeichnet. Nachdem der Sender eine negative Quittung eines Empfängers erhalten hat, wird eine Übertragungswiederholung an die gesamte Multicast-Gruppe eingeleitet. Zu beachten ist, dass empfängerinitiierte Protokolle nichtdeterministisch sind, da der Sender nicht entscheiden kann, ob respektive zu welchem Zeitpunkt, alle Empfänger die Nachricht korrekt empfangen haben. Der Sender muss folglich Nachrichten unendlich lange speichern, um auch späte

Anforderungen für Übertragungswiederholungen befriedigen zu können oder aber die Entscheidung treffen, dass vermutlich die Nachricht gelöscht werden kann. Das in Abschnitt 2.5.3 besprochene MTP [Armstrong et al. 1992] gehört zur hier vorgestellten Klasse K-NACK.

### 4.2.3 Empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-NAPP)

Empfängerinitiierte Protokolle können, im Gegensatz zur Protokollklasse K-NACK, negative Quittungen auch mittels eines Multicast-Diensts an die gesamte Gruppe adressieren. Andere Empfänger, die ebenfalls von diesem Nachrichtenverlust betroffen sind, unterdrücken nun ihre eigene negative Quittung, nachdem sie eine andere empfangen haben. Damit nicht zu viele Empfänger gleichzeitig eine negative Quittung senden, wird das Senden um eine zufällige Zeit verzögert [Ramakrishnan & Jain 1987]. Nachdem der Sender eine negative Quittung erhalten hat, wird eine Übertragungswiederholung ausgelöst. Wiederum wird die Annahme getroffen, dass die Übertragungswiederholung an die gesamte Gruppe übertragen wird. Dies ist hier sogar notwendig, da ein Multicast-NACK für eine beliebige Anzahl von Gruppenmitgliedern sprechen kann. Die Kurzbezeichnung NAPP ist eine in der Literatur übliche Bezeichnung für diese Protokollklasse und steht für „NACK with NACK suppression“ [Levine & Garcia-Luna-Aceves 1998]. Als Beispiel für ein NAPP-Protokoll wurde in Abschnitt 2.5.3 SRM [Floyd et al. 1997] besprochen.

### 4.2.4 Hierarchisches senderinitiiertes Protokoll (K-HACK)

Für alle hierarchischen Protokolle wird angenommen, dass die Empfänger einen Kontrollbaum bilden, mit dem Sender an der Wurzel. Der Kontrollbaum kann mit einem beliebigen Verfahren aufgebaut werden, allerdings kann nur TRS aus dem vorigen Kapitel Skalierbarkeit für große Teilnehmerzahlen bieten. Ein Empfänger bestätigt den korrekten Empfang einer Datennachricht mit einer positiven Quittung an seinen Vaterknoten im Kontrollbaum. Für diese Protokollklasse wird die Annahme getroffen, dass eine Übertragungswiederholung an eine lokale Gruppe vom Gruppenführer mittels eines Multicast-Diensts gesendet wird. Dazu kann eine eigene Multicast-Gruppe für alle Kindknoten gebildet werden, um den Sendebereich entsprechend einzuschränken. Eine weitere Möglichkeit besteht darin, an die globale Multicast-Gruppe zu senden und den Auslieferungsbereich durch Beschränkung der Teilstreckenanzahl (TTL) einzugrenzen. In Abschnitt 2.5.3 wurde das RMTP-Protokoll [Paul et al. 1997] besprochen, welches den Auslieferungsbereich mittels „subcasting“ einschränkt, was mit herkömmlichen Routern nicht möglich ist. Sieht man davon ab, so kann RMTP zu der hier vorgestellten Klasse K-HACK gezählt werden.

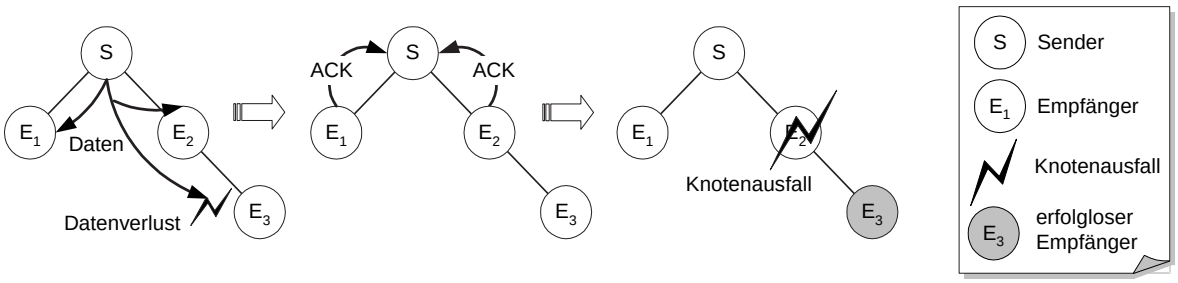



*Abbildung 4.2: Szenario für Datenverlust ohne aggregierte Quittungen*

### 4.2.5 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-HNAPP)

Protokolle, die zur K-HNAPP-Klasse gehören, sind ebenfalls hierarchisch und verwenden negative Quittungen mit Quittungsunterdrückung. Damit ist ebenfalls das Problem des Nichtdeterminismus gegeben wie es in Abschnitt 4.2.2 bereits beschrieben wurde. Für ein hierarchisches Protokoll lässt sich dieses Problem zumindest einfacher beheben als für flache Protokolle, ohne die Skalierbarkeit maßgeblich zu reduzieren. Dazu werden von allen Empfängern periodische positive Quittungen an den Vaterknoten übermittelt. Diese werden nicht für Übertragungswiederholungen ausgewertet, sondern um festzustellen, welche Datennachrichten bereits von allen Empfängern empfangen wurden und somit bei den Gruppenführern und schließlich auch beim Sender als oberster Gruppenführer gelöscht werden können. TMTP [Yavatkar et al. 1995] ist ein Beispiel für ein Protokoll der Klasse K-HNAPP (siehe Abschnitt 2.5.3).

### 4.2.6 Hierarchisches senderinitiiertes Protokoll mit aggregierten Quittungen (K-HAACK)

Die Protokollklasse K-HAACK beinhaltet hierarchische Protokolle, die neben normalen positiven Quittungen auch aggregierte positive Quittungen verwenden. Bevor näher erläutert wird, was aggregierte Quittungen sind, soll durch die folgende Problemstellung die Grundlage für das Verständnis deren Notwendigkeit gelegt werden.

In Abbildung 4.2 ist die Auslieferung einer Multicast-Nachricht veranschaulicht. Die linke Seite der Abbildung zeigt, dass nur die Empfänger $E_1$ und $E_2$ die Nachricht erhalten und eine positive Quittung (mittleres Bild) an den Sender übermitteln. Nach einer geeigneten Wartezeit wird $E_2$ als Gruppenführer von $E_3$ dessen Nachrichtenverlust bemerken und eine Übertragungswiederholung initiieren. Angenommen, in diesem Moment fällt $E_2$ aus, genauer gesagt, nachdem er seine positive Quittung an den Sender übermittelt hat und bevor er die Übertragungswiederholung für $E_3$ durchführt. In dieser

Situation passiert folgendes: Sowohl $S$ als auch $E_1$ können keine Übertragungswiederholung für $E_3$ durchführen, da $E_1$ seine Nachricht bereits gelöscht haben kann und $S$ ebenfalls, bedingt durch den Erhalt beider ACK-Nachrichten, den Speicherplatz freigegeben hat. Somit ist für $E_3$ die Nachricht endgültig verloren. Zusammenfassend lässt sich sagen, dass alle bisher beschriebenen Protokollklassen keine Zuverlässigkeitsgarantie bei einem Knotenausfall geben können.

Die Protokollklasse K-HAACK kann durch die Verwendung von aggregierten Quittungen auch bei einem Knotenausfall die Nachrichtenauslieferung an alle verbleibenden Empfänger sicherstellen. Dazu wird der Fehlerbehebungsmechanismus von dem Speicherfreigabemechanismus getrennt. Wie bereits in Protokollklasse K-ACK werden positive Quittungen an den Vaterknoten gesendet, der beim Ausbleiben dieser eine Übertragungswiederholung initiiert. Das Eintreffen der positiven Quittungen von allen Kindknoten führt indes nicht zur Löschung der betreffenden Nachricht und Speicherfreigabe. Dafür sind die aggregierten positiven Quittungen zuständig, die von einem Empfänger erst dann an den Vaterknoten übermittelt werden, nachdem von allen Kindknoten bereits eine aggregierte Quittung erhalten wurde und auch die Datennachricht selbst empfangen wurde. Die folgende Beschreibung definiert das Protokollverhalten im Detail:

1. Ein Blattknoten sendet eine aggregierte Quittung an den Vaterknoten, nachdem eine Datennachricht erfolgreich empfangen wurde. Blattknoten sind Empfänger im Kontrollbaum, die nicht als Gruppenführer fungieren.

2. Ein Gruppenführer sendet eine positive Quittung an den Vaterknoten, nachdem eine Datennachricht erfolgreich empfangen wurde.

3. Ein Gruppenführer wartet eine definierte Zeit auf das Eintreffen der positiven Quittungen aller Kindknoten. Das Eintreffen einer aggregierten Quittung wird ebenfalls als positive Quittung gewertet. Tritt eine Zeitüberschreitung auf, ohne dass von allen Kindknoten eine positive oder aggregierte Quittung empfangen wurde, so wird die Nachricht wiederholt.

4. Ein Gruppenführer wartet eine definierte Zeit auf das Eintreffen der aggregierten Quittungen aller Kindknoten. Nach dem Eintreffen kann die entsprechende Datennachricht aus dem Sendepuffer gelöscht werden. Eine eigene aggregierte Quittung an den Vaterknoten zeigt an, dass der gesamte Teilbaum, mit dem Gruppenführer als Wurzel, die Datennachricht empfangen hat. Tritt hingegen eine Zeitüberschreitung beim Warten auf die aggregierten Quittungen auf, so wird eine Quittungsnachforderung an die betreffenden Kindknoten mittels Unicast versendet und erneut gewartet, bis alle aggregierten Quittungen eingetroffen sind oder erneut Quittungsnachforderungen versendet.

5. Empfängt ein Knoten Übertragungswiederholungen oder Quittungsnachforderungen nachdem eine aggregierte Quittung gesendet wurde oder nachdem die Bedingungen zum Senden einer

aggregierten Quittung erfüllt sind, so wird eine aggregierte Quittung anstatt einer positiven Quittung an den Vaterknoten gesendet.

Wie der Definition entnommen werden kann wird nicht festgelegt was passiert, wenn ein Knoten tatsächlich ausfällt. Dies bleibt dem konkreten Transportprotokoll überlassen und ist für die folgende Analyse nicht von Bedeutung. Es ist ausreichend zu spezifizieren, welche Vorkehrungen getroffen werden müssen, um einen Knotenausfall behandeln zu können und nicht, wie die Behandlung im Einzelnen ausfällt. RMTP II [Whetten & Taskale 2000] ist ein Beispiel für ein Protokoll, das aggregierte Quittungen verwendet.

Eine weitere Besonderheit dieser Protokollklasse ist, dass abhängig von der Anzahl ausstehender Quittungen, Multicast oder Unicast für die Übertragungswiederholungen Verwendung findet. Dazu vergleicht der Gruppenführer die Anzahl ausstehender Quittungen mit einem definierten Schwellwert. Wird dieser unterschritten, so wird Unicast anstatt Multicast für die Übertragungswiederholungen verwendet. Die dargelegte Erweiterung könnte auch in die Protokollklasse K-HACK integriert werden. Um die Vergleichbarkeit mit verwandten Arbeiten in diesem Gebiet zu gewährleisten, wurde dies nicht durchgeführt (siehe [Pingali et al. 1994] und [Levine & Garcia-Luna-Aceves 1998]). Für Protokolle mit aggregierten Quittungen gibt es hingegen keine klassifizierenden und analysierenden Vorarbeiten.

### 4.2.7 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung und aggregierten Quittungen (K-HANAPP)

Analog zur vorigen Protokollklasse lassen sich auch negative Quittungen mit Quittungsunterdrückung und aggregierte Quittungen kombinieren, die nun anstelle der periodischen Quittungen von Protokollklasse K-HNAPP eingesetzt werden. Für ein empfängerinitiiertes Protokoll ist dies nur sinnvoll, wenn es periodisch geschieht, da ansonsten die Skalierbarkeit darunter leidet und das Verhalten sich dem von Protokollklasse K-HAACK annähern würde. Die folgende Beschreibung definiert das detaillierte Protokollverhalten:

1. Entdeckt ein Empfänger den Verlust oder fehlerhaften Empfang einer Nachricht, so initiiert dieser eine negative Quittung mittels eines Multicast-Dients an die lokale Gruppe oder an die globale Gruppe mit geeigneter Begrenzung des Auslieferungsbereichs. Die negative Quittung wird indes nicht sofort gesendet, sondern um eine zufällige Zeit verzögert. Trifft vor dem Sendezeitpunkt die negative Quittung eines anderen Empfängers der lokalen Gruppe ein, so wird die eigene negative Quittung unterdrückt. Sollte nach einer geeigneten Wartezeit die Datennachricht nicht eintreffen, dann wird der Vorgang wiederholt.

2. Empfängt ein Gruppenführer eine negative Quittung aus seiner lokalen Gruppe, so sendet er eine Übertragungswiederholung mittels eines Multicast-Diensts an die lokale Gruppe oder an die globale Gruppe mit geeigneter Begrenzung des Auslieferungsbereichs.

3. Nach einer definierten Anzahl erfolgreich empfangener Datennachrichten sendet ein Blattknoten im Kontrollbaum eine aggregierte Quittung an seinen Vaterknoten.

4. Ein Gruppenführer sendet selbst eine aggregierte Quittung an seinen Vaterknoten, nachdem er von allen Kindknoten eine aggregierte Quittung erhalten hat und selbst die entsprechende Datennachricht empfangen hat. Zu diesem Zeitpunkt kann auch die Datennachricht aus dem Sendepuffer entfernt werden. Sind nach einer geeigneten Wartezeit nicht alle aggregierten Quittungen eingetroffen, sendet der Gruppenführer eine Quittungsnachforderung mittels Unicast an die entsprechenden Kindknoten, deren aggregierte Quittung fehlt.

5. Empfängt ein Knoten eine Quittungsnachforderung, nachdem eine aggregierte Quittung gesendet wurde oder nachdem die Bedingungen zum Senden einer aggregierten Quittung erfüllt sind, so wird eine aggregierte Quittung an den Vaterknoten gesendet.

## 4.3 Verwandte Arbeiten

Die Leistungsbewertung von Protokollen ist mittels Analyse, Simulation oder Messungen in realen Umgebungen möglich. Alle Möglichkeiten haben sowohl Vor– als auch Nachteile und wurden für Multicast-Protokolle bereits praktiziert. Simulationen wurden beispielsweise von [Floyd et al. 1997, Lehman et al. 1998, Hashimoto et al. 1999] durchgeführt und Messungen von [Lin & Paul 1996, Hofmann 1998]. Die in der vorliegenden Arbeit durchgeführten Analysen haben gegenüber Simulationen und Messungen den Vorteil, nachprüfbar zu sein und auch für sehr große Multicast-Gruppen Ergebnisse zu liefern.

In einer großen Anzahl von Veröffentlichungen wird ein neues Broadcast– [Chang & Maxemchuk 1984] oder Multicast-Protokoll [Shiroshita et al. 1996, Gu & Garcia-Luna-Aceves 1997, Costello & McCanne 1999] oder zumindest eine neue Methode, Übertragungswiederholungen durchzuführen [Mohan et al. 1988], vorgestellt und anschließend eine Analyse dessen präsentiert. Schließlich wird in [Bhagwat et al. 1994] ausgehend von einem Multicast-Protokoll der Einfluss des Spannbaums untersucht. Bei der hier vorgestellten Analyse werden stattdessen generische Klassen von Multicast-Protokollen untersucht und anschließend miteinander verglichen. Verwandte Arbeiten in diesem Gebiet werden nun ausführlich vorgestellt. Dabei wird unterschieden zwischen der CPU–, der Bandbreiten– und der Verzögerungsanalyse.

### 4.3.1 Analysen zur CPU-Belastung

Die erste vergleichende Analyse von generischen zuverlässigen Multicast-Protokollen wurde von [Pingali et al. 1994] vorgenommen. Analysiert wurden drei flache Protokollklassen, deren Definition übernommen wurde und in Abschnitt 4.2 als K-ACK, K-NACK und K-NAPP vorgestellt wurden. Die Autoren nehmen an, dass die Bandbreite des Netzwerks schneller wächst als die CPU-Leistung und sehen daher in der CPU-Leistung die kritische Ressource. Ausgehend von einer Betrachtung des Verarbeitungsaufwands der Daten- und Quittungsnachrichten bei Sender und Empfängern wird der maximal zu erzielende Durchsatz bestimmt. Pingali et al. schlussfolgern aus ihren Ergebnissen, dass die empfängerinitiierte Fehlerbehebung der Protokollklassen K-NACK und K-NAPP gegenüber K-ACK einen verbesserten Durchsatz und eine verbesserte Skalierbarkeit bietet. Die Quittungsunterdrückung sichert der K-NAPP-Klasse schließlich den höchsten Durchsatz und die beste Skalierbarkeit.

In [Levine & Garcia-Luna-Aceves 1998] wurde die Arbeit von Pingali et al. aufgegriffen und wesentlich erweitert. Untersucht wurden ringbasierte und hierarchische Protokollklassen. Ringbasierte Protokolle [Chang & Maxemchuk 1984, Whetten et al. 1994] besitzen Vorteile, wenn die Ordnungserhaltung der Nachrichtenauslieferung oder zumindest eine Totalordnung gewünscht ist. Für die Skalierbarkeit erbringen sie keine Verbesserung und würden entsprechend der hier gegebenen Klassifikation den flachen Protokollklassen zugeordnet werden. Ihr wesentlicher Nachteil ist das aufwändige Ring-Management. Sie werden daher im weiteren Verlauf dieser Arbeit nicht weiter behandelt. Die hierarchischen Protokolle wurden in zwei Klassen eingeteilt, deren Definition in Abschnitt 4.2 als K-HACK und K-HNAPP übernommen wurde. Wesentliches Merkmal bei deren Definition von Levine et al. war, dass diese Klassen im Gegensatz zu K-NACK und K-NAPP deterministisch arbeiten und dadurch mit endlichem Speicher auskommen. Das Hauptergebnis dieser Arbeit ist die Erkenntnis, dass Kontrollbäume die Skalierbarkeit für sehr große Gruppen garantieren können. Als prinzipielles Problem der Protokolle mit Quittungsunterdrückung wird angeführt, dass diese eine aufwändige Verwaltung der Zeitgeber bedingen, welche in den Analysen nicht betrachtet wird und eventuell die Skalierbarkeit beeinträchtigen. Auch hierfür sind Kontrollbäume bestens geeignet, da die akkurate Bestimmung der Zeitgeber überwiegend nur innerhalb lokaler Gruppen notwendig ist.

In zwei Veröffentlichungen wird der Einfluss spezieller Mechanismen auf die CPU-Belastung untersucht. In [Kasera et al. 1997] wurden die Protokollklassen aus [Pingali et al. 1994] erweitert. Anstatt alle Übertragungswiederholungen an dieselbe Multicast-Gruppe zu senden, werden dafür mehrere Multicast-Gruppen verwendet. Eine Übertragungswiederholung wird in Abhängigkeit von der Sequenznummer an eine spezielle Multicast-Gruppe gesendet. Alle Empfänger, die Nachrichtenverlust bemerkt haben, melden sich an die entsprechenden Multicast-Gruppen an. Damit wird verhindert, dass jeder Empfänger alle Übertragungswiederholungen empfangen muss. Dies führt zu einer Entlastung der Empfänger und zu einem verbesserten Durchsatz, bedingt andererseits aber auch höheren Aufwand auf der Vermittlungsschicht durch häufige Gruppenein– und –austritte.

In [Nonnenmacher et al. 1997] wurde erörtert, inwieweit Forward-Error-Correction (FEC) in Kombination mit Quittungen und Übertragungswiederholungen den Durchsatz verbessern kann. Dabei wurden sowohl unabhängige Verluste als auch die Übertragung über einen binären Spannbaum mit den entsprechenden räumlichen Verlustabhängigkeiten betrachtet (siehe auch Abschnitt 4.4). Nonnenmacher et al. kommen zu dem Ergebnis, dass FEC zu weniger Übertragungswiederholungen führt und die Skalierbarkeit verbessert wird.

### 4.3.2 Analysen zum Bandbreitenbedarf

Analysen zum Bandbreitenbedarf wurden parallel von mehreren Autoren durchgeführt, jeweils mit unterschiedlicher Ausrichtung. Die ersten zwei vorgestellten Arbeiten beziehen ihre Berechnung des Bandbreitenbedarfs auf ein spezielles Netzwerkmodell. Eine Nachricht wird dabei über genau drei Verbindungen weitergeleitet, eine Verbindung vom Sender zum Backbone, eine Verbindung im Backbone und eine Verbindung vom Backbone zum Empfänger. Damit kann der gesamte Bandbreitenbedarf im Netzwerk bestimmt werden.

In [Kasera et al. 1998] wurden generische hierarchische Protokolle untersucht und mit flachen Verfahren verglichen. Bei den hierarchischen Protokollen wurde unterschieden, ob ein Gruppenführer ein normaler Empfänger ist oder ein spezieller Server, der im Netzwerk in unmittelbarer Nähe von zentralen Routern platziert ist. Das wichtigste Ergebnis ist, dass hierarchische Protokolle weniger Bandbreite verbrauchen und einen höheren Durchsatz erzielen als flache Verfahren. Innerhalb der hierarchischen Verfahren hilft die strategisch günstige Ausrichtung der Gruppenführer im Server-Ansatz, die besten Ergebnisse zu erzielen.

Die Kombination von Übertragungswiederholungen mit FEC wurde in [Nonnenmacher et al. 1998] untersucht. Grundlage ihrer Analysen sind zwei hierarchische und ein flaches Protokoll, wobei nur ein hierarchisches Protokoll kein FEC verwendet. Nonnenmacher et al. schließen aus den Ergebnissen, dass das hierarchische Protokoll mit FEC den geringsten Bandbreitenbedarf verursacht.

Eine weitere interessante Arbeit über flache Protokollklassen haben [Poo & Goscinski 1998] vorgelegt. Anstatt nur stop-and-wait Übertragungswiederholungen zu untersuchen, wurden hier auch selective-repeat und go-back-N Verfahren betrachtet. Untersucht wurde nicht der Bandbreitenbedarf, sondern die Bandbreiteneffizienz, d.h. wie gut ein Übertragungswiederholungsschema die verfügbare Bandbreite nutzen kann. Erwartungsgemäß führt die selective-repeat Variante zur besten Bandbreiteneffizienz.

### 4.3.3 Analysen zur Nachrichtenverzögerung

Die ersten Verzögerungsanalysen von generischen zuverlässigen Multicast-Protokollen wurden von [Yamamoto et al. 1997] und [DeCleene 1996] präsentiert. In beiden Artikeln werden flache Verfahren

untersucht. DeCleene hat sich besonders für das Verhalten der Protokolle in unterschiedlichen Netzwerken interessiert. Untersucht wurden Stern– und Lineartopologien. Im Rahmen der vorliegenden Arbeit ist die Arbeit von Yamamoto et al. von größerem Interesse. Die untersuchten Protokollklassen sind identisch mit [Pingali et al. 1994]. Unter der Berücksichtigung von Bearbeitungsverzögerungen auf einem Knoten und der Übertragungsverzögerung im Netzwerk, wird die durchschnittliche Verzögerung vom Senden einer Nachricht bis zur Auslieferung analysiert. Ein Ergebnis dieser Arbeit ist, dass die Quittungsunterdrückung der Klasse K-NAPP zu einer erhöhten Nachrichtenverzögerung führen kann. Dennoch ist die wichtigste Aussage, dass die Klasse K-NAPP den geringsten Bearbeitungsaufwand auf den Knoten verursacht und daher als einzige ausreichende Skalierbarkeit bei flachen Protokollklassen bieten kann.

Neben dem Bandbreitenbedarf ist in der bereits vorgestellten Arbeit von Nonnenmacher et al. [Nonnenmacher et al. 1998] auch die Nachrichtenverzögerung analysiert worden. Allerdings ist die Verzögerungsanalyse sehr knapp gehalten und nicht vollständig. Der Artikel basiert auf [Lacher 1998], in dem eine vollständige Analyse beschrieben wird. Im Gegensatz zu den oben vorgestellten Arbeiten wird dabei ein äußerst einfaches Modell vorausgesetzt. So wird das Warteschlangenverhalten auf einem Knoten beim Bearbeiten von Nachrichten nicht im Detail betrachtet und die Quittungsbearbeitung wird gänzlich vernachlässigt. Neben der Bandbreitenanalyse ergibt auch die Verzögerungsanalyse einen Vorteil für das hierarchische Protokoll mit FEC.

### 4.3.4 Diskussion und Abgrenzung

Die Ausführungen zu verwandten Arbeiten zeigen, dass bereits sehr interessante Analysen durchgeführt und wichtige Ergebnisse erzielt wurden. Hervorzuheben sind vor allem die Arbeiten von [Pingali et al. 1994] und [Yamamoto et al. 1997] die von anderen Autoren häufig als Grundlage für eigene Erweiterungen herangezogen wurden. Das wichtigste Resümee aus allen Arbeiten ist, dass hierarchische Protokolle die beste Skalierbarkeit bieten.

Im Rahmen dieser Arbeit ergibt sich dennoch eine wichtige Fragestellung, die in vorangegangenen Arbeiten keine Berücksichtigung gefunden hat. Wenn hierarchische Protokolle die beste Alternative darstellen, wie müssen diese dann konfiguriert werden, um die optimale Leistung zu erbringen, d.h. wie sieht ein optimaler Verzweigungsgrad aus? In [Kasera et al. 1998] ist diese Problemstellung bereits angedacht, jedoch nicht umfassend analysiert worden. Die nachfolgende Analyse verfolgt als ein wichtiges Ziel, einen optimalen Verzweigungsgrad zu bestimmen der mittels des TRS-Diensts beim Kontrollbaumaufbau konfiguriert werden kann. Folgende weitere Untersuchungen wurden in bisherigen Arbeiten noch nicht durchgeführt und lassen ebenfalls gänzlich neue Erkenntnisse erwarten:

- Erstmals finden Protokollklassen mit aggregierten Quittungen zur Behandlung von Knotenausfall Berücksichtigung.

- Die Bandbreitenanalyse untersucht die Bandbreitenanforderung an den Endsystemen und nicht innerhalb stark vereinfachter Netzwerkmodelle.
- Hierarchische Multicast-Protokolle werden einer umfassenden Verzögerungsanalyse unterzogen.
- Neben der durchschnittlichen Auslieferungsverzögerung wird auch die Umlaufverzögerung bis zum Empfang aller Quittungen und die maximale Auslieferungsverzögerung betrachtet (siehe Definition in Abschnitt 4.5.4).

Daneben werden zusätzlich folgende Erweiterungen vorgenommen, die in früheren Arbeiten nicht berücksichtigt wurden und wesentlich realistischere Ergebnisse für alle Protokollklassen erwarten lassen:

- Neben der Möglichkeit des Datenverlusts wird auch die Möglichkeit des Quittungsverlusts betrachtet.
- Die Quittungsunterdrückung resultiert nur im Idealfall in genau einer negativen Quittung pro verlorener Datennachricht, was einzig mittels synchronisierter Uhren und unrealistisch hohen Wartezeiten erreicht werden kann. Die Analyse berücksichtigt, dass auch mehrere negative Quittungen gesendet werden können.
- Es wird berücksichtigt, dass Multicast-Übertragungswiederholungen des Gruppenführers auch Empfänger außerhalb der eigenen lokalen Gruppe erreichen können.

Schließlich werden zur Überprüfung der Resultate im letzten Abschnitt Simulationsergebnisse präsentiert, was ebenfalls in bisherigen Arbeiten noch nicht erfolgte.

## 4.4 Modell

In diesem Abschnitt wird das der Protokollanalyse zugrundeliegende System– und Fehlermodell vorgestellt. Vorausgesetzt wird ein asynchrones System aus autonomen Rechnerknoten, die durch ein Kommunikationsnetzwerk miteinander verbunden sind. Eine nichtleere Teilmenge der Rechnerknoten bildet eine Multicast-Gruppe. Alle Teilnehmer der Multicast-Gruppe arbeiten zuverlässig und ausschließlich für das analysierte Multicast-Transportprotokoll. Eine Überlegung hinsichtlich stabilem oder flüchtigem Speicher erübrigt sich bei zuverlässig arbeitenden Knoten. Innerhalb der Multicast-Gruppe wird mittels Nachrichtenaustausch kommuniziert und synchronisiert. Der Nachrichtenaustausch unterliegt temporären Kommunikationsfehlern, d.h. in endlicher Zeit ist Kommunikation zwischen zwei belieben Rechnerknoten möglich.

Es wird angenommen, dass ein einzelner Sender eine Multicast-Nachricht mittels eines unzuverlässigen Multicast-Diensts an die Gruppe von $R$ Empfängern sendet. Der Sender ist selbst Mitglied der Gruppe. Mit einer Wahrscheinlichkeit $q_D$ wird die Multicast-Nachricht von einem beliebigen Empfänger nicht korrekt empfangen. Alle Empfänger sind identisch, d.h. haben die selbe Nachrichtenverlustwahrscheinlichkeit. Mittels einer Quittung teilt der Empfänger dem Sender anschließend seinen Zustand über den Nachrichtenempfang mit. Die Quittung kann ihrerseits vom Sender nicht korrekt empfangen werden. Die Wahrscheinlichkeiten dafür sind $p_A$ für positive Unicast-Quittungen, $p_N$ für negative Quittungen die per Unicast gesendet werden und $q_N$ für negative Quittungen, die per Multicast gesendet werden. Der Analyse liegt die Annahme zugrunde, dass Übertragungsfehler nur temporärer Natur sind, d.h. keine Rechner dauerhaft unerreichbar sind. Nachrichtenverluste müssen jedoch nicht zwangsläufig einzelne Nachrichten betreffen, sondern können auch gebündelt auftreten.

Die Wahrscheinlichkeit für Nachrichtenverlust definiert das Systemmodell nur zum Teil. Eine weitere, doch untergeordnete Eigenschaft ist die Abhängigkeit der Verluste in temporaler und räumlicher Dimension. In [Yajnik et al. 1996] und [Yajnik et al. 1999] wurde die temporale und räumliche Abhängigkeit der Verluste im MBone sehr detailliert untersucht. Die Autoren schließen aus Messungen, dass der Zeitraum für temporal abhängige Nachrichtenverluste kleiner als eine Sekunde ist. Dies bedeutet, dass alle Nachrichten, die zeitlich mehr als eine Sekunde auseinanderliegen, keine temporale Abhängigkeit bezüglich ihrer Verlustwahrscheinlichkeiten aufweisen. Der Verlust einer Nachricht kann immerhin von dem Verlust anderer Nachrichten innerhalb der vergangenen Sekunde abhängen. Doch selbst bei Nachrichtenverlusten innerhalb einer Zeitspanne von einer Sekunde wurde festgestellt, dass die überwiegende Anzahl von Verlusten einzelne Verluste sind und damit ebenfalls keine Abhängigkeiten besitzen. Obwohl sich häufig mehrere Empfänger gleiche Router-Pfade teilen, wurde bei räumlichen Abhängigkeiten nur eine sehr kleine Korrelation der Verluste ermittelt. Eine wichtige Ausnahme schränkt diese Aussage hingegen ein. Die Verluste auf der ersten Verbindung des Senders, zum Beispiel über eine Modem-Verbindung zum Router des Internet-Providers, werden von allen Empfängern gleichermaßen erfahren und sind somit räumlich stark korreliert.

Aus diesen Messungen lässt sich folgendes schließen. Die Annahme, dass Nachrichtenverluste temporal unabhängig sind, entspricht weitestgehend der Realität und führt nur bei Nachrichtenverlusten innerhalb einer Sekunde zu Ungenauigkeiten in der Analyse. Diese Ungenauigkeiten sind aber recht gering, da ebenso Nachrichtenverluste innerhalb einer Sekunde überwiegend temporal unkorreliert sind. Deshalb wird für die folgende Analyse die Annahme getroffen, dass Nachrichtenverluste temporal unabhängig sind. Im ersten Teil der Analyse wird zudem angenommen, dass keine räumlichen Abhängigkeiten für den Nachrichtenverlust bestehen. Dies vereinfacht die Analyse und erlaubt die Unabhängigkeit von einer konkreten Annahme über die Netzstruktur. Dadurch sind die Erkenntnisse sowohl für kleinere Netze wie einzelne LANs oder Intranetze als auch für Weitverkehrsnetze, wie dem Internet, gültig. Der dadurch akzeptierten Ungenauigkeit über die Verlustabhängigkeit auf Verbindun-

gen nahe dem Sender soll in Abschnitt 4.5.5 mit einem erweiterten aber spezialisierten Systemmodell, das diesen Fall berücksichtigt, begegnet werden.

Die Protokollleistung in Anbetracht von Knotenausfällen zu untersuchen ist nicht Ziel dieser Abhandlung. Auch in der Literatur wird dieser Fall nicht erörtert. Eine derartige Analyse ist schwierig, da die Fehlersemantik bei Knotenausfall schwer zu definieren ist (siehe Abschnitt 2.5.1) und für viele reale Protokolle das Verhalten bei Knotenfehlern nicht hinreichend definiert wurde. Wurde die Fehlersemantik definiert, dann unterscheiden sich die Semantiken beträchtlich, so dass eine Klassenbildung kaum möglich ist. Erschwerend kommt hinzu, dass viele Protokolle Knotenausfälle nicht behandeln können. Der wichtigste Grund, Knotenausfall nicht zu berücksichtigen, ist allerdings durch deren geringen Wahrscheinlichkeit im Vergleich zum Nachrichtenverlust begründet. Eine Berücksichtigung von Knotenausfällen würde die Ergebnisse lediglich mäßig beeinflussen. Vor allem ist nicht zu erwarten, dass einzelne Klassen dadurch merklich bevorzugt oder benachteiligt werden, was den relativen Vergleich der Klassen nicht verändert. Folgende Beobachtungen sollen dennoch aufgestellt werden:

- Ein temporärer Knotenausfall mit anschließender Wiederherstellung des vollen Zustands vor dem Ausfall ist in der Analyse identisch mit dem Verlust aller an ihn in dieser Zeit gesendeten Nachrichten. Dieser Fall kann somit durch eine entsprechend erhöhte Wahrscheinlichkeit für den Nachrichtenverlust eingeschlossen werden.

- Ein temporärer Knotenausfall ohne vollständige Wiederherstellung des Zustands vor dem Ausfall bedingt die Verwendung eines Protokolls der Klasse K-HAACK oder K-HANAPP. Nur diese beiden Klassen können bei diesem Fehlermodell die Zustellung aller Nachrichten zusichern. Auch dieser Fall kann durch eine entsprechend erhöhte Wahrscheinlichkeit für den Nachrichtenverlust eingeschlossen werden.

- Ein permanenter Knotenausfall bedingt ebenfalls die Verwendung von aggregierten Quittungen und damit ein Protokoll der Klasse K-HAACK oder K-HANAPP. Allerdings sind zusätzliche Mechanismen notwendig, um einen permanent ausgefallenen Knoten aus der Empfängermenge zu entfernen, um nicht für unbegrenzte Zeit Übertragungswiederholungen senden zu müssen. Da solche Mechanismen hier nicht definiert wurden, ist ein permanenter Knotenausfall durch die folgende Analyse nicht abgedeckt. Zu beachten ist, dass dieser Fall identisch zu einem permanenten Nachrichtenverlust ist, z.B. hervorgerufen durch eine permanente Netzpartitionierung (siehe oben).

Obwohl die beiden ersten Fälle prinzipiell durch die Analyse abgedeckt sind, muss eine wichtige Einschränkung gemacht werden. Der Wiederanlauf eines Knotens nach dem Ausfall benötigt eine gewisse Zeit in der keine Nachrichten empfangen werden können. Dies führt dazu, dass der Nachrichtenverlust nicht sporadisch auftritt, sondern eine temporale Abhängigkeit besitzt, die in der Analyse keine Berücksichtigung findet.

## 4.5 Protokollanalyse

Die vorgestellte Protokollanalyse umfasst den Bandbreitenbedarf, die CPU-Belastung und die Nachrichtenverzögerung. Ausgangspunkt aller Analysen ist die detaillierte Betrachtung des verwendeten Fehlerbehebungsmechanismus und der daraus resultierenden Anzahl notwendiger Übertragungswiederholungen bei Nachrichtenverlust. Die notwendige Anzahl Übertragungswiederholungen ist damit der wichtigste Einflussparameter für alle Ergebnisse, da viele andere Größen, wie die Anzahl der zu verarbeitenden Daten– und Quittungsnachrichten und die Zeitgeberunterbrechungen, davon abhängen. Da die meisten dieser Quantitäten wiederholt sowohl für die Bandbreiten–, die CPU– als auch die Verzögerungsanalyse benötigt werden, soll deren Berechnung nun vorab erfolgen.

### 4.5.1 Bestimmung der notwendigen Übertragungen

Die folgenden Ergebnisse wurden erstmals in [Maihöfer et al. 2000] und [Maihöfer 2000a] vorgestellt. Die Anzahl notwendiger Übertragungen kann aus der Sichtweise eines einzelnen Empfängers oder der gesamten Gruppe betrachtet werden. Die Berechnungsmethode wird erst allgemein vorgestellt, bevor in den weiteren Abschnitten detailliert auf alle Protokollklassen eingegangen wird.

Die durchschnittliche Anzahl notwendiger Übertragungen, bis zum korrekten Empfang einer Datennachricht bei einem einzelnen Empfänger $r$, wird als $M_r$ bezeichnet. Für die gesamte Gruppe wird die Abkürzung $M$ gewählt. Wichtigster Einflussparameter auf $M_r$ und $M$ ist die Wahrscheinlichkeit für eine Übertragungswiederholung. Diese Wahrscheinlichkeit für Übertragungswiederholungen soll vorerst allgemein mit $\tilde{p}$ bezeichnet werden. In Tabelle 4.1 sind die wichtigsten Abkürzungen zusammengefasst.

In den folgenden Abschnitten wird die Berechnung von $\tilde{p}$ dargestellt, die von der konkreten Protokollklasse abhängig ist. $\tilde{p}$ ist die Wahrscheinlichkeit für eine notwendige Übertragungswiederholung aus der Sichtweise eines einzelnen Empfängers. Dies kann durch ein Bernoulli-Experiment beschrieben werden [Rüegg 1986, S. 49]. Ein Bernoulli-Experiment liefert nur zwei mögliche Ergebnisse: Erfolg und Misserfolg. Die Wahrscheinlichkeit für Erfolg ist beschrieben durch:

$$1 - \tilde{p}. \tag{4.1}$$

Eine Folge von unabhängigen Bernoulli-Experimenten mit jeweils gleicher Erfolgswahrscheinlichkeit $p$ wird durch die geometrische Verteilung beschrieben [Rüegg 1986, S. 50]:

$$P(X = k) \quad = \quad (1-p)^{k-1} p \qquad \text{mit } k = 1, 2, \ldots \tag{4.2}$$

| | | |
|---|---|---|
| $R$ | - | Anzahl der Empfänger einer Multicast-Gruppe ohne den Sender |
| $B$ | - | Verzweigungsgrad des Kontrollbaums bzw. Größe einer lokalen Gruppe |
| $p_D, q_D$ | - | Wahrscheinlichkeit für Datenverlust bei Übertragungen mittels Unicast oder Multicast |
| $p_A, p_N$ | - | Wahrscheinlichkeit für den Verlust von positiven oder negativen Quittungen mittels Unicast |
| $q_N$ | - | Wahrscheinlichkeit für den Verlust von negativen Quittungen mittels Multicast |
| $\tilde{p}$ | - | Wahrscheinlichkeit, dass eine Übertragungswiederholung notwendig ist |
| $M_r$ | - | Anzahl notwendiger Übertragungen für einen beliebigen Empfänger $r$, um eine Datennachricht erfolgreich zu empfangen |
| $M$ | - | Anzahl notwendiger Übertragungen für alle Empfänger, um eine Datennachricht erfolgreich zu empfangen |

*Tabelle 4.1: Notation zur Analyse der Transportprotokolle*

$X$ ist die Anzahl der Experimente, die durchgeführt werden müssen, bis zum ersten Mal ein Erfolg eintritt. Bezogen auf die hier interessierende Situation ist $X$ die Anzahl Übertragungen, bis zum ersten erfolgreichen Empfang einer Nachricht bei einem einzelnen Empfänger. Der Erwartungswert der geometrischen Verteilung ist wie folgt gegeben [Rüegg 1986, S. 56]:

$$E(X) = \frac{1}{p}. \tag{4.3}$$

Mit $p = 1 - \tilde{p}$ (siehe Formel 4.1) ergibt sich der Erwartungswert für die Anzahl der Übertragungen für einen einzelnen Empfänger zu:

$$E(M_r) = \frac{1}{1-\tilde{p}}. \tag{4.4}$$

Diese Größe ist später erforderlich zur Bestimmung der Quittungsanzahl, die ein einzelner Empfänger versendet. Es verbleibt noch die Bestimmung der notwendigen Übertragungen für die gesamte Gruppe. Die Wahrscheinlichkeit, dass die Anzahl der Übertragungen für einen einzelnen Empfänger $M_r$ kleiner oder gleich $m$ ist ($m = 1, 2, ...$), ist gegeben durch:

$$P(M_r \leq m) = 1 - \tilde{p}^m. \tag{4.5}$$

$\tilde{p}^m$ ist die Wahrscheinlichkeit, dass eine Nachricht m-mal verloren geht, wovon die komplementäre Wahrscheinlichkeit zu bilden ist, um die interessierende Größe zu erhalten. Um die Gesamtwahrscheinlichkeit zu erreichen, müssen die Einzelwahrscheinlichkeiten aller Empfänger multipliziert werden. Unter der Annahme, dass $R$ Empfänger zur Multicast-Gruppe gehören, ergibt sich:

$$P(M \leq m) \quad = \quad \prod_{r=1}^{R} P(M_r \leq m) = (1 - \tilde{p}^m)^R. \tag{4.6}$$

Mit Hilfe der Binomischen Formel lässt sich diese Darstellung umformen. Die Binomische Formel lautet [Rüegg 1986, S. 25]:

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}. \tag{4.7}$$

Mit $a = \tilde{p}^m$, $b = -1$ und $n = R$ ergibt sich:

$$(a+b)^n = (-1)^R (1-\tilde{p}^m)^R. \tag{4.8}$$

Mit Hilfe der Formeln 4.7 und 4.8 ergibt sich für Formel 4.6 folgende alternative Darstellung:

$$P(M \leq m) = (-1)^R \sum_{i=0}^{R} \binom{R}{i} (-1)^{R-i} (\tilde{p}^m)^i. \tag{4.9}$$

Auf den ersten Blick scheint diese Formel eine umständlichere Darstellung zu sein, aber sie erlaubt im weiteren Verlauf wichtige Vereinfachungen. Das alternierende Glied $(-1)^{R-i}$ kann mit $(-1)^R$ zu $(-1)^i$ vereinfacht werden. Aus der obigen Formel kann $P(M = m)$ bestimmt werden:

$$\begin{aligned} P(M=m) &= P(M \leq m) - P(M \leq m-1) \\ &= \sum_{i=0}^{R} \binom{R}{i} (-1)^i \tilde{p}^{im} - \sum_{i=0}^{R} \binom{R}{i} (-1)^i \tilde{p}^{i(m-1)} \\ &= \sum_{i=0}^{R} \binom{R}{i} (-1)^i \left( \tilde{p}^{im} - \tilde{p}^{i(m-1)} \right) \\ &= \sum_{i=0}^{R} \binom{R}{i} (-1)^i \tilde{p}^{im} (1-\tilde{p}^{-i}). \end{aligned} \tag{4.10}$$

Der Erwartungswert ist definiert als Summe über alle Wertebelegungen multipliziert mit deren Eintrittswahrscheinlichkeiten [Rüegg 1986, S. 53]. Für die Anzahl Übertragungen ergibt sich somit:

$$E(M) = \sum_{m=1}^{\infty} m P(M=m) \tag{4.11}$$

$$= \sum_{m=1}^{\infty} m \left( \sum_{i=0}^{R} \binom{R}{i} (-1)^i \tilde{p}^{im} (1-\tilde{p}^{-i}) \right). \tag{4.12}$$

Ausgehend von der Überlegung $\sum_{m=1}^{\infty} a_m (\sum_{i=0}^{R} b_i) = a_1 b_0 + a_1 b_1 + \ldots + a_2 b_0 + a_2 b_1 + \ldots = b_0 a_1 + b_0 a_2 + \ldots = \sum_{i=0}^{R} (b_i \sum_{m=1}^{\infty} a_m)$ lassen sich die beiden Summen vertauschen:

$$\begin{aligned} E(M) &= \sum_{i=0}^{R} \left( \binom{R}{i} (-1)^i (1-\tilde{p}^{-i}) \sum_{m=1}^{\infty} m \tilde{p}^{im} \right) \\ &= \sum_{i=1}^{R} \left( \binom{R}{i} (-1)^i (1-\tilde{p}^{-i}) \sum_{m=1}^{\infty} m \tilde{p}^{im} \right) + 0, \text{ für } i = 0. \end{aligned} \tag{4.13}$$

Die weitere Vereinfachung dieser Formel bezieht sich auf die zweite Summe. Diese lässt sich auch folgendermaßen beschreiben:

$$
\begin{array}{rcl}
 & & \tilde{p}^{1i} + \\
 & & \tilde{p}^{2i} + \tilde{p}^{2i} + \\
1\tilde{p}^{1i} + 2\tilde{p}^{2i} + 3\tilde{p}^{3i} + \ldots & = & \tilde{p}^{3i} + \tilde{p}^{3i} + \tilde{p}^{3i} + \\
 & & \underbrace{\vdots}_{s_1} + \underbrace{\vdots}_{s_2} + \underbrace{\vdots}_{s_3} + \underbrace{\ddots}_{s_\ldots}
\end{array} \tag{4.14}
$$

$$
s_1 = \sum_{x=1}^{\infty} (\tilde{p}^i)^x = \tilde{p}^i \sum_{x=0}^{\infty} (\tilde{p}^i)^x \tag{4.15}
$$

$$
s_2 = \sum_{x=2}^{\infty} (\tilde{p}^i)^x = \tilde{p}^{2i} \sum_{x=0}^{\infty} (\tilde{p}^i)^x \tag{4.16}
$$

$$
s_n = \sum_{x=n}^{\infty} (\tilde{p}^i)^x = \tilde{p}^{ni} \sum_{x=0}^{\infty} (\tilde{p}^i)^x \tag{4.17}
$$

Die einzelnen Zeilen mit $\sum_{x=0}^{\infty} (\tilde{p}^i)^x$ sind geometrische Reihen die nach [Merziger & Wirth 1993, S. 337] gegen $\frac{1}{1-\tilde{p}^i}$ konvergieren:

$$
s_1 = \tfrac{\tilde{p}^i}{1-\tilde{p}^i}, \quad s_2 = \tfrac{\tilde{p}^{2i}}{1-\tilde{p}^i}, \quad s_n = \tfrac{\tilde{p}^{ni}}{1-\tilde{p}^i} \tag{4.18}
$$

Formel 4.14 lässt sich damit wie folgt als Summe der einzelnen Zeilen aufschreiben, was wiederum eine geometrische Reihe ergibt:

$$
s_1 + s_2 + \ldots = \frac{1}{1-\tilde{p}^i} \sum_{x=1}^{\infty} (\tilde{p}^i)^x = \frac{1}{1-\tilde{p}^i} \tilde{p}^i \sum_{x=0}^{\infty} (\tilde{p}^i)^x = \frac{\tilde{p}^i}{1-\tilde{p}^i} \frac{1}{1-\tilde{p}^i} = \frac{\tilde{p}^i}{(1-\tilde{p}^i)^2}. \tag{4.19}
$$

Nun kann die ursprüngliche Formel 4.13 mit Hilfe der Überlegungen aus Formel 4.19 vereinfacht werden zu:

$$
\begin{aligned}
E(M) &= \sum_{i=1}^{R} \binom{R}{i} (-1)^i (1 - \tilde{p}^{-i}) \frac{\tilde{p}^i}{(1-\tilde{p}^i)^2} \\
&= \sum_{i=1}^{R} \binom{R}{i} (-1)^i (-1) \frac{1-\tilde{p}^i}{(1-\tilde{p}^i)^2} \\
&= \sum_{i=1}^{R} \binom{R}{i} (-1)^{i+1} \frac{1}{1-\tilde{p}^i}.
\end{aligned} \tag{4.20}
$$

Die nun erhaltene Formel für den Erwartungswert der Anzahl von Übertragungen ist eine endliche Reihe und folglich numerisch berechenbar.

Zusammenfassend wurde bisher die Berechnung der notwendigen Übertragungen zum fehlerfreien Empfang einer Nachricht bei einem beliebigen Empfänger, $M_r$, und bei allen Empfängern, $M$, erörtert.

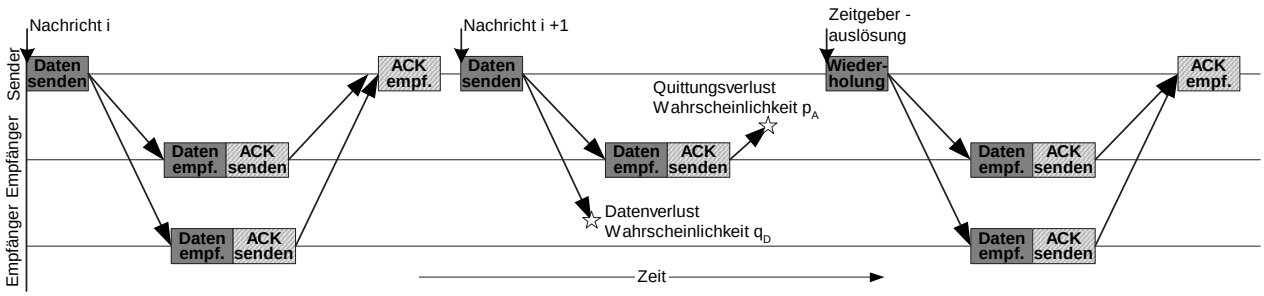



*Abbildung 4.3: Übertragungswiederholungen bei senderinitiierten Protokollen*

In den folgenden Abschnitten wird die einzige Unbekannte der Berechnung, die Fehlerwahrscheinlichkeit für Übertragungen, $\tilde{p}$, in Abhängigkeit der konkreten Protokollklassen bestimmt. Darüber hinaus werden die weiteren, von der Anzahl Übertragungen abhängigen, Quantitäten analysiert, insoweit sie für die Bandbreiten–, CPU– oder Verzögerungsanalyse benötigt werden. Dazu gehört im Besonderen die Anzahl Quittungsmeldungen, sowohl gesendete als auch empfangene.

#### 4.5.1.1 Senderinitiiertes Protokoll (K-ACK)

Bei einem senderinitiierten Protokoll ist der Sender für die Erkennung von Datenverlust zuständig. Der Sender startet eine Übertragungswiederholung aufgrund einer oder mehrerer ausbleibenden positiven Quittungen. In Abschnitt 4.2.1 wurde bereits beschrieben, dass der Sender dabei nicht zwischen Datenverlust oder Verlust der Quittung unterscheiden kann. In Abbildung 4.3 ist dies anschaulich dargestellt. Die Fehlerwahrscheinlichkeit $\tilde{p}$ setzt sich somit aus der Nachrichtenverlustwahrscheinlichkeit für Daten, $q_D$, und positive Quittungen, $p_A$, zusammen:

$$\tilde{p} = q_D + (1 - q_D)p_A. \tag{4.21}$$

Damit ist die notwendige Anzahl von Übertragungen für Protokollklasse K-ACK bereits vollständig bestimmt. Jede erfolgreich empfangene Übertragung führt bei einem Empfänger zum Senden einer positiven Quittung. Die Gesamtanzahl der positiven Quittungen, $\widetilde{L}^{ACK}$, die der Sender erhält, ist:

$$E(\widetilde{L}^{ACK}) \quad = \quad RE(M^{ACK})(1 - q_D)(1 - p_A). \tag{4.22}$$

Das heißt, der Sender erhält für jede Datenübertragung ($M^{ACK}$) und von jedem Empfänger ($R$) eine Quittung, vorausgesetzt weder die Datenübertragung $(1 - q_D)$ noch die Quittung $(1 - p_A)$ geht verloren. Das hochgestellte ACK in $M^{ACK}$ soll verkürzt für die Protokollklasse K-ACK stehen. Allgemein kennzeichnet die Tilde über einer Zufallsvariable *empfangene* Nachrichten. Im Anhang sind alle verwendeten Abkürzungen zusammengefasst.

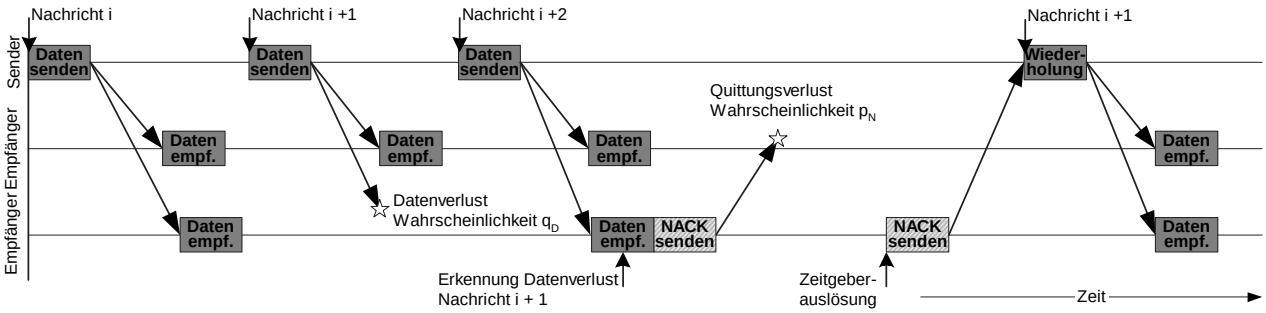



*Abbildung 4.4: Übertragungswiederholungen bei empfängerinitiierten Protokollen*

#### 4.5.1.2 Empfängerinitiiertes Protokoll (K-NACK)

Für ein empfängerinitiiertes Protokoll ist nur der eigentliche Datenverlust für Übertragungswiederholungen verantwortlich, da die Entscheidung von den Empfängern getroffen wird (siehe Abbildung 4.4). Es gibt somit keine unnötigen Übertragungen durch Quittungsverluste. Die Fehlerwahrscheinlichkeit $\tilde{p}$ ergibt sich zu:

$$\tilde{p} = q_D. \tag{4.23}$$

Für empfängerinitiierte Protokolle ergibt sich dagegen ein anderes Problem. Durch den Verlust von negativen Quittungen ist es möglich, dass Empfänger auf Übertragungswiederholungen warten, die der Sender nicht ausführen wird. Aus der Sichtweise des Empfängers wird nach einer angemessenen Wartezeit eine weitere negative Quittung gesendet, da entweder die vorige verloren ging oder die Übertragungswiederholung erneut nicht empfangen wurde. Das heißt, aus der Sichtweise des Empfängers gibt es eine größere oder zumindest gleich große Anzahl scheinbarer Übertragungswiederholungen als aus der Sichtweise des Senders tatsächlich durchgeführt wurden. Diese scheinbaren Übertragungen aus der Sichtweise des Empfängers werden als Runden bezeichnet. Die Rundenanzahl für einen einzelnen Empfänger $O_r^{NACK}$ bzw. für alle Empfänger $O^{NACK}$ ist gegeben durch:

$$E(O^{NACK}) = E(M^{NACK}) + E(O_e^{NACK}) \tag{4.24}$$

$$E(O_r^{NACK}) = E(M_r^{NACK}) + E(O_{e,r}^{NACK}), \tag{4.25}$$

wobei $O_e$ und $O_{e,r}$ leere Runden bezeichnen, d.h. Runden in denen alle negativen Quittungen verloren gehen und folglich keine Übertragungswiederholung initiiert wird. Um die Anzahl leerer Runden zu bestimmen wird folgendermaßen vorgegangen. Beginnend mit der Bestimmung der Anzahl gesendeter negativer Quittungen lässt sich die Wahrscheinlichkeit bestimmen, dass alle negativen Quittungen verloren gehen. Schließlich wird der Erwartungswert über diese Wahrscheinlichkeit berechnet.

$N_k$ bezeichnet die Anzahl negativer Quittungen, die in Runde $k$ gesendet werden:

$$E(N_k) = {q_D}^k R. \tag{4.26}$$

${q_D}^k$ ist die Wahrscheinlichkeit für einen einzelnen Empfänger, dass bis zur Runde $k$ alle Datenübertragungen oder negative Quittungen verloren gehen und aus diesem Grund eine neue negative Quittung gesendet wird. Die Wahrscheinlichkeit, dass alle gesendeten negativen Quittungen in Runde $k$ verloren gehen, ist damit:

$$p_k = {p_N}^{E(N_k)} = {p_N}^{{q_D}^k R}. \tag{4.27}$$

Nun kann der Erwartungswert der leeren Runden, $E(O_e^{NACK})$, bestimmt werden. Er setzt sich zusammen aus den leeren Runden nach der ersten und allen folgenden Übertragungen. Diese werden wiederum durch ein Bernoulli-Experiment und durch die geometrische Verteilung beschrieben:

$$E(O_e^{NACK}) = \sum_{k=1}^{E(M^{NACK})-1} \left(\frac{1}{1-p_k} - 1\right) \tag{4.28}$$

$$E(O_{e,r}^{NACK}) = \sum_{k=1}^{E(M_r^{NACK})-1} \left(\frac{1}{1-p_k} - 1\right). \tag{4.29}$$

$(1/1-p_k)$ ist dabei der Erwartungswert für die Anzahl von Versuchen eine negative Quittung zu senden bis zum ersten Erfolg (vgl. Formel 4.4). Der erste Erfolg wird vom Erwartungswert subtrahiert. Somit bleibt die Anzahl der leeren Runden pro Übertragung übrig. Die gesamte Anzahl negativer Quittungen, die in allen Übertragungsrunden gesendet werden, ist gegeben durch:

$$E(L^{NACK}) = \sum_{k=1}^{E(M^{NACK})} E(N_k)\frac{1}{1-p_k}. \tag{4.30}$$

Damit ist die Gesamtanzahl negativer Quittungen, $\widetilde{L}^{NACK}$, die vom Sender empfangen werden:

$$E(\widetilde{L}^{NACK}) = \sum_{k=1}^{E(M^{NACK})} E(N_k)\frac{1}{1-p_k}(1-p_N). \tag{4.31}$$

#### 4.5.1.3 Empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-NAPP)

Analog zur Protokollklasse K-NACK ist auch K-NAPP ein empfängerinitiiertes Protokoll. Die Fehlerwahrscheinlichkeit für Übertragungen $\tilde{p}$ ist ebenfalls bestimmt durch die Wahrscheinlichkeit für den Datenverlust $q_D$. Die Anzahl der Übertragungen $M^{NAPP}$ und $M_r^{NAPP}$ werden analog zur Berechnung in K-NACK ermittelt.

Die Anzahl leerer Runden nach der $k$-ten Übertragung ist auch hier durch die Fehlerwahrscheinlichkeit $p_k$ gegeben. $p_k$ ist die Wahrscheinlichkeit, dass alle gesendeten negativen Quittungen in Runde $k$ verloren gehen:

$$p_k = {q_N}^{E(N_k)}. \tag{4.32}$$

Die Berechnung von $N_k$ unterscheidet sich von der vorigen Protokollklasse erheblich, da in Protokollklasse K-NAPP negative Quittungen unterdrückt werden und im Idealfall nur eine gesendet wird. Die Gesamtanzahl gesendeter negativer Quittungen setzt sich wie folgt zusammen:

- Negative Quittung des ersten Empfängers, der die Datennachricht nicht erhalten hat.
- Negative Quittung eines zweiten Empfängers, der weder die Datennachricht noch die Quittung des ersten, nicht erfolgreichen Empfängers erhalten hat.
- Analog für den dritten und alle weiteren nicht erfolgreichen Empfänger.

Damit ist die Gesamtanzahl gesendeter negativer Quittungen:

$$
\begin{aligned}
E(N_k) &= \textstyle\sum_{i=1}^{R} E(N_{k,i}) && (4.33)\\
E(N_{k,1}) &= {q_D}^k && (4.34)\\
E(N_{k,2}) &= {q_D}^k\Big(1 - E(N_{k,1}) + E(N_{k,1})q_N\Big)\\
&= E(N_{k,1})\Big(1 - E(N_{k,1}) + E(N_{k,1})q_N\Big)\\
&= E(N_{k,1}) - E(N_{k,1})^2 + E(N_{k,1})^2 q_N && (4.35)\\
E(N_{k,n}) &= E(N_{k,n-1}) - E(N_{k,n-1})^2 + E(N_{k,n-1})^2 q_N \quad , n > 1. && (4.36)
\end{aligned}
$$

${q_D}^k$ ist die Wahrscheinlichkeit für den ersten Empfänger, dass er eine negative Quittung in Runde $k$ senden wird. Der zweite Empfänger sendet eine negative Quittung, wenn die Datennachricht verloren ging (${q_D}^k$) und die negative Quittung des ersten Empfängers verloren ging ($N_{k,1}q_N$) oder der erste Empfänger keine negative Quittung gesendet hat ($1 - N_{k,1}$). Für alle weiteren Empfänger gilt dies analog, woraus schließlich die Formel 4.36 abgeleitet wird.

Die vorige Berechnung zur Anzahl negativer Quittungen stützt sich einzig auf die Wahrscheinlichkeiten, dass außer Datennachrichten auch die negativen Quittungen verloren gehen können und dadurch unnötige weitere negative Quittungen von Empfängern gesendet werden. Dies ist eine vereinfachte Annahme, da durch die Nachrichtenlaufzeiten im Netzwerk simultanes, mehrfaches Senden und dadurch ebenfalls zusätzliche negative Quittungen auftreten können. Infolgedessen werden die Formeln 4.34 und 4.36 mit der Wahrscheinlichkeit für simultanes Senden von negativen Quittungen, $p_s$, erweitert zu:

$$
\begin{aligned}
E(N_{k,1}) &= {q_D}^k && (4.37)\\
E(N_{k,n}) &= E(N_{k,n-1}) - E(N_{k,n-1})^2 + E(N_{k,n-1})^2(q_N + p_s - q_N p_s),\ n > 1. && (4.38)
\end{aligned}
$$

Nun kann die Anzahl leerer Runden bestimmt werden (vgl. Formel 4.24 und 4.28):

$$
\begin{aligned}
E(O_e^{NAPP}) &= \textstyle\sum_{k=1}^{E(M^{NAPP})-1}\left(\frac{1}{1-p_k} - 1\right) && (4.39)\\
E(O_{e,r}^{NAPP}) &= \textstyle\sum_{k=1}^{E(M_r^{NAPP})-1}\left(\frac{1}{1-p_k} - 1\right). && (4.40)
\end{aligned}
$$

Abschließend soll aus der Gesamtzahl gesendeter Quittungen, $L^{NAPP}$, noch die Gesamtanzahl der Quittungen, $\widetilde{L}^{NAPP}$, bestimmt werden, die beim Sender empfangen werden:

$$E(\widetilde{L}^{NAPP}) = E(L^{NAPP})(1-q_N) \quad (4.41)$$

$$E(L^{NAPP}) = \sum_{k=1}^{E(M^{NAPP})} E(N_k)\frac{1}{1-p_k}. \quad (4.42)$$

$N_k$, die Anzahl gesendeter Quittungen pro Übertragung und $p_k$, die Fehlerwahrscheinlichkeit, dass alle negativen Quittungen verloren gehen, wurden in Formel 4.33 und Formel 4.32 bestimmt. $\frac{1}{1-p_k}$ ist der Erwartungswert der geometrischen Verteilung (siehe Formel 4.4); in dieser Situation somit die Anzahl leerer Runden, in denen alle negativen Quittungen verloren gehen inklusive der ersten erfolgreichen Quittung.

Die Berechnung der Nachrichtenanzahl eines Empfängers erfordert weitergehende Überlegungen aufgrund dessen, dass durch die Quittungsunterdrückung nicht mehr jeder erfolglose Empfänger eine Quittung sendet. Zuerst soll die durchschnittliche Anzahl negativer Quittungen, $L_\phi^{NAPP}$, bestimmt werden, die in einer Runde gesendet werden:

$$E(L_\phi^{NAPP}) = \frac{E(L^{NAPP})}{E(O^{NAPP})}. \quad (4.43)$$

$L^{NAPP}$, die Gesamtzahl gesendeter Quittungen wurde bereits in Formel 4.42 berechnet. In ähnlicher Weise wie $L^{NAPP}$ wird die durchschnittliche Anzahl erfolgloser Empfänger bestimmt, die eine negative Quittung senden wollen:

$$E(R_N) = \frac{1}{E(O^{NAPP})}\sum_{k=1}^{E(M^{NAPP})} {q_D}^k R\frac{1}{1-p_k}. \quad (4.44)$$

${q_D}^k R$ ist die Anzahl erfolgloser Empfänger in Übertragungsrunde $k$. $p_k$, die Fehlerwahrscheinlichkeit, dass alle negativen Quittungen verloren gehen, wurde in Formel 4.32 berechnet. $\frac{1}{1-p_k}$ ist die Anzahl leerer Runden, in denen alle negativen Quittungen verloren gehen inklusive des ersten Erfolgs.

#### 4.5.1.4 Hierarchisches senderinitiiertes Protokoll (K-HACK)

Hierarchische Protokolle organisieren ihre Empfänger in einem Kontrollbaum mit lokalen Gruppen. Alle Übertragungswiederholungen werden nur für eine lokale Gruppe durchgeführt. Deshalb ist bei der Bestimmung der notwendigen Übertragungswiederholungen auch nur eine beispielhafte lokale Gruppe von Interesse. Eine derartige lokale Gruppe weist viele Ähnlichkeiten mit einem flachen senderinitiierten Protokoll auf. Aufgrund dieser Ähnlichkeit kann die Bestimmung der Übertragungsanzahl aus der Klasse H-ACK abgeleitet werden. Der wesentliche Unterschied besteht darin, dass

nicht mehr alle Empfänger, $R$, sondern nur noch die Mitglieder der lokalen Gruppe, $B$, für eine Übertragungswiederholung verantwortlich sind. $B$ ist identisch zu dem in Abschnitt 3.2.1 eingeführten Verzweigungsgrad (engl. Branching Factor). Daher ist $M^{HACK}$ gegeben durch (vgl. Formel 4.20):

$$E(M^{HACK}) = \sum_{i=1}^{B} \binom{B}{i} (-1)^{i+1} \frac{1}{1-\tilde{p}^i}. \tag{4.45}$$

$\tilde{p}$ ist in Formel 4.21 und $M_r^{HACK}$ ist in Formel 4.4 gegeben. Die Anzahl positiver Quittungen, $E(\widetilde{L}^{HACK})$, die der Sender erhält, ist (vgl. Formel 4.22):

$$E(\widetilde{L}^{HACK}) = BE(M^{HACK})(1-q_D)(1-p_A). \tag{4.46}$$

Um den Sendebereich von Übertragungswiederholungen auf die Mitglieder der lokalen Gruppe zu beschränken gibt es zwei Möglichkeiten. Entweder bekommt jede lokale Gruppe eine eigene Multicast-Adresse zugewiesen oder die Vermittlung der Nachricht wird durch den TTL-basierten Ausbreitungsbereich von IP-Paketen eingeschränkt [Postel 1981b]. Während die zuerst genannte Möglichkeit sehr geeignet erscheint, ergeben sich bei genauerer Betrachtung dennoch einige Nachteile. Sicherlich ist der erhöhte Bedarf an Multicast-Adressen — diese stellen eine endliche Ressource dar (vgl. Abschnitt 2.3.2) — ein derartiger Nachteil. Der hauptsächliche Nachteil ergibt sich indessen aus dem Mehraufwand auf der Vermittlungsschicht. Für jede Multicast-Gruppe muss ein zusätzlicher Spannbaum aufgebaut werden (siehe Abschnitt 2.4.1). Um dies zu umgehen wird beispielsweise von TMTP [Yavatkar et al. 1995] der Sendebereich durch geeignete Begrenzung der IP-TTL beschränkt. Eine Steuerung durch die IP-TTL ist keineswegs perfekt, d.h. das Paket wird auch von weiteren Knoten außerhalb der eigenen lokalen Gruppe, aber innerhalb des IP-TTL-Radius, empfangen. Dies führt zu zusätzlichen Nachrichten, was sich in einem erhöhten Bandbreitenbedarf und einer erhöhten CPU-Belastung auswirkt. Direkt vom Sender oder Vaterknoten im Kontrollbaum erhält ein Empfänger die folgenden Übertragungen:

$$E(\widetilde{N}_r^{HACK}) = E(M^{HACK})(1-q_D). \tag{4.47}$$

Auf diese Übertragungen muss mit einer positiven Quittung reagiert werden. Dennoch empfängt ein beliebiger Empfänger insgesamt:

$$E(\widetilde{N}_{r,t}^{HACK}) = E(M^{HACK})(1-q_D) + (G-1)\Big(E(M^{HACK})-1\Big)(1-q_D)p_l \tag{4.48}$$

Übertragungen. Die Berechnung der Anzahl Gruppenführer $G$, erfolgt in Formel 4.52. Davon wird die eigene Gruppe subtrahiert. Von der Anzahl Übertragungen, $M^{HACK}$, wird die initiale Übertragung subtrahiert, um nur die Übertragungswiederholungen zu erhalten. $p_l$ ist die Wahrscheinlichkeit eine Multicast-Nachricht zu erhalten, die nicht für die eigene lokale Gruppe bestimmt ist. Dies soll im Folgenden Lokalgruppenüberlappung genannt werden.

Für einen Gruppenführer ergibt sich, abweichend von obiger Formel, die Gesamtanzahl zu empfangener Nachrichten zu:

$$E(\widetilde{N}_g^{HACK}) \quad = \quad E(M^{HACK})(1-q_D)+(G-2)\Big(E(M^{HACK})-1\Big)(1-q_D)p_l. \tag{4.49}$$

Ein Gruppenführer empfängt nur von $G-2$ weiteren lokalen Gruppen Übertragungen, da er selbst Mitglied von zwei lokalen Gruppen ist. Eine lokale Gruppe wird aus seinen Kindknoten gebildet und eine weitere wird aus seinen Geschwisterknoten und dem Vaterknoten gebildet.

Es verbleibt noch die Berechnung der Anzahl lokaler Gruppen, $G$. Die Höhe des Wurzelknotens im Kontrollbaum sei definiert als eins und die Höhe aller weiterer Knoten als die des Vaterknotens erhöht um eins. Mit dieser Definition folgt die Höhe des Kontrollbaums aus der Anzahl der Empfänger $R = 1-B^h/1-B$ (siehe Formel 3.19 auf Seite 85) zu:

$$h \quad = \quad \log_B\Big(R(B-1)+1\Big). \tag{4.50}$$

Für spätere Berechnungen ist zusätzlich die durchschnittliche Höhe eines Empfängers im Kontrollbaum, $\overline{h}$, von Bedeutung:

$$\overline{h} = \frac{\Big(\sum_{i=1}^{h-2}(i+1)*B^i\Big)+\Big(R-\sum_{j=1}^{h-2}B^j\Big)h}{R}. \tag{4.51}$$

Aus der Kontrollbaumhöhe lässt sich die Anzahl der lokalen Gruppen bestimmen:

$$G \quad = \quad \sum_{i=0}^{h-2} B^i. \tag{4.52}$$

#### 4.5.1.5 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-HNAPP)

Jede lokale Gruppe der Protokollklasse K-HNAPP ist identisch zur flachen Protokollklasse K-NAPP. Deshalb ist die Anzahl der notwendigen Übertragungen vom Verzweigungsgrad $B$ abhängig und wird nach Formel 4.45 mit $\tilde{p} = q_D$ berechnet. Auch die Anzahl Runden, $O^{HNAPP}$ und $O_r^{HNAPP}$, werden entsprechend den Formeln 4.39 und 4.40 berechnet.

Die Reaktion eines Empfängers auf eine nicht empfangene Übertragung ist das Senden einer negativen Quittung mittels Multicast an die lokale Gruppe. Dabei ergeben sich mit der Einschränkung des Auslieferungsbereichs auf die lokale Gruppe die gleichen Probleme wie sie bereits zu Protokollklasse K-HACK für Übertragungswiederholungen ausgeführt wurden. Für die Anzahl der empfangenen negativen Quittungen bei einem beliebigen Gruppenführer gilt somit:

$$E(\widetilde{L}^{HNAPP}) \quad = \quad E(L^{HNAPP})(1-q_N)+(G-1)E(L^{HNAPP})(1-q_N)p_l \tag{4.53}$$

$$E(L^{HNAPP}) \quad = \quad \sum_{k=1}^{E(M^{HNAPP})} E(N_k)\tfrac{1}{1-p_k}. \tag{4.54}$$

$L^{HNAPP}$ sind die gesendeten negativen Quittungen innerhalb einer lokalen Gruppe. Der erste Teil der Formel 4.53 sind die empfangenen negativen Quittungen, die von der eigenen lokalen Gruppe gesendet wurden. Unter der Annahme, dass insgesamt $G$ lokale Gruppen existieren, werden auch potentiell von den weiteren $G-1$ lokalen Gruppen die Quittungen empfangen. Eine Lokalgruppenüberlappung findet mit Wahrscheinlichkeit $p_l$ statt. $N_k$, die Anzahl der gesendeten negativen Quittungen pro Runde $k$ und $p_k$, die Fehlerwahrscheinlichkeit für leere Runden werden analog zu den Formeln 4.32 und 4.33 bis 4.38 mit $B$ anstatt $R$ berechnet.

Die Klasse K-HNAPP erweitert die Klasse K-NAPP nicht ausschließlich um den Kontrollbaum, sondern auch um die periodischen positiven Quittungen, die den Nichtdeterminismus allgemeiner empfängerinitiierter Protokolle aufhebt. Die Anzahl periodischer Quittungen $S$, die bei einem Gruppenführer empfangen werden, ist:

$$E(S) = (1-p_A)B. \tag{4.55}$$

Damit ist die Anzahl der Übertragungen und Quittungen für die Protokollklasse K-HNAPP vollständig bestimmt.

#### 4.5.1.6 Hierarchisches senderinitiiertes Protokoll mit aggregierten Quittungen (K-HAACK)

Die Protokollklasse K-HAACK sendet Übertragungswiederholungen entweder per Unicast oder per Multicast, abhängig von der Anzahl Empfänger, deren positive Quittung ausbleibt. Unter der Annahme, dass $p_t$ die Wahrscheinlichkeit angibt die Übertragungswiederholung mit Unicast zu versenden, ist die Anzahl der Unicast, $M_u^{HAACK}$, und Multicast, $M_m^{HAACK}$, Übertragungswiederholungen gegeben durch:

$$E(M_u^{HAACK}) = p_t\Big(E(M^{HAACK})-1\Big) \tag{4.56}$$

$$E(M_m^{HAACK}) = (1-p_t)\Big(E(M^{HAACK})-1\Big)+1. \tag{4.57}$$

Dabei ist zu beachten, dass die erste Übertragung stets mit Multicast erfolgt. Zur Berechnung von $M^{HAACK}$ ist es erforderlich, die Fehlerwahrscheinlichkeit zu berechnen:

$$\tilde{p} = \underbrace{\overbrace{p_t p_D}^{Unicast} + \overbrace{(1-p_t)q_D}^{Multicast}}_{Datenverlust} + \underbrace{\Big[1-\Big(\overbrace{p_t p_D}^{Unicast} + \overbrace{(1-p_t)q_D}^{Multicast}\Big)\Big]p_A}_{kein\ Datenverlust\ aber\ Quittungsverlust}. \tag{4.58}$$

$q_D$ ist die Wahrscheinlichkeit für Datenverlust bei Multicast-Übertragungen und $p_D$ die entsprechende Wahrscheinlichkeit für Unicast-Übertragungen. $\tilde{p}$ eingesetzt in Formel 4.45 ergibt schließlich

$M^{HAACK}$. Um $p_t$, die Wahrscheinlichkeit dass Unicast für eine Übertragungswiederholung verwendet wird, zu berechnen, wird ein Schwellwert $\phi$ eingeführt der konfigurierbar ist. Ist die Anzahl der Knoten, die eine Übertragungswiederholung benötigen, kleiner als $\phi$, dann wird Unicast verwendet und ansonsten Multicast. $p_t$ ist nun gegeben durch:

$$p_t = \frac{1}{E(M^{HAACK})} \sum_{k=1}^{E(M^{HAACK})} \begin{cases} 1 & , Bq_D{}^k < \phi \\ 0 & , Bq_D{}^k \geq \phi \end{cases} \tag{4.59}$$

Leider ist dies eine zyklische Definition der Berechnungen, da $p_t$ verwendet wird um $M^{HAACK}$ zu berechnen und umgekehrt. Um zu einem Ergebnis zu kommen, muss angenommen werden, dass die Wahrscheinlichkeit für Datenverlust bei Multicast-Übertragungen, $q_D$, gleich der Wahrscheinlichkeit für Unicast-Übertragungen, $p_D$, ist.

Zu beachten ist, dass $M_u^{HAACK}$ nur die Anzahl der Unicast-Übertragungsrunden, nicht die tatsächliche Anzahl gesendeter Nachrichten angibt, da pro Runde mehr als eine Unicast-Datennachricht gesendet werden kann. Deswegen wird die durchschnittliche Empfängeranzahl $N_u$ bestimmt, die pro Runde eine Unicast-Übertragungswiederholung erfordern:

$$E(N_u) = \frac{1}{E(M_u^{HAACK})} \sum_{k=1}^{E(M^{HAACK})} \begin{cases} Bq_D{}^k & , Bq_D{}^k < \phi \\ 0 & , Bq_D{}^k \geq \phi \end{cases} \tag{4.60}$$

Aus diesen Ergebnissen lässt sich die Gesamtanzahl der Übertragungen, $E(\widetilde{N}_r^{HAACK})$, ermitteln, die einen beliebigen Empfänger, $r$, mittels Unicast oder Multicast vom Vaterknoten im Kontrollbaum erreicht:

$$E(\widetilde{N}_r^{HAACK}) = \frac{E(N_u)}{B} E(M_u^{HAACK})(1-p_D) + E(M_m^{HAACK})(1-q_D). \tag{4.61}$$

Ein Empfänger $r$ empfängt nicht nur die Übertragungswiederholungen des Vaterknotens, sondern eventuell auch Übertragungswiederholungen von Gruppenführern anderer lokaler Gruppen, die in $\widetilde{N}_{r,t}^{HAACK}$ berücksichtigt sind:

$$E(\widetilde{N}_{r,t}^{HAACK}) = E(\widetilde{N}_r^{HAACK}) + (G-1)\Big(E(M_m^{HAACK}) - 1\Big)(1-q_D)p_l. \tag{4.62}$$

Die Anzahl von Übertragungen, die mit einer positiven bzw. aggregierten Quittung bestätigt werden, sind:

$$E(L_a^{HAACK}) = p_c E(\widetilde{N}_r^{HAACK}) \tag{4.63}$$

$$E(L_{aa}^{HAACK}) = (1-p_c)E(\widetilde{N}_r^{HAACK}), \tag{4.64}$$

d.h. nur Übertragungen der eigenen lokalen Gruppe werden quittiert. $p_c$ ist die Wahrscheinlichkeit, dass bei den Kindknoten ein Problem oder eine Verzögerung auftritt und folglich keine aggregierte

Quittung, sondern nur eine normale, positive Quittung gesendet werden kann. Die Anzahl eintreffender positiver Quittungen bei einem Gruppenführer ist:

$$E(\widetilde{L}_a^{HAACK}) = B(1-p_A)E(L_a^{HAACK}). \tag{4.65}$$

Nachdem die Datennachrichten alle bestätigt wurden kann es vorkommen, dass aufgrund von Quittungsverlust aggregierte Quittungen fehlen. Diese werden durch Quittungsnachfragen, die der Gruppenführer an die betreffenden Kindknoten mittels Unicast sendet, nachgefordert. Die Anzahl der Quittungsnachforderungsrunden $O_q^{HAACK}$ ist durch die Wahrscheinlichkeit $\hat{p}$, dass eine Anfrage fehlschlägt, bestimmt:

$$\hat{p} = p_q + (1-p_q)p_{AA}. \tag{4.66}$$

$p_{AA}$ ist die Wahrscheinlichkeit für den Verlust von aggregierten Quittungen und $p_q$ ist die Wahrscheinlichkeit für den Verlust von Quittungsnachforderungen. $O_q^{HAACK}$ kann nun mittels Formel 4.20 ermittelt werden:

$$E(O_q^{HAACK}) = \sum_{i=1}^{B_{aa}} \binom{B_{aa}}{i} (-1)^{i+1} \frac{1}{1-\hat{p}^i}. \tag{4.67}$$

$B_{aa}$ ist die Anzahl Empfänger, die der Gruppenführer beim ersten Auftreten einer Zeitgeberunterbrechung für die aggregierten Quittungen abfragen muss:

$$E(B_{aa}) = B\Big(p_c + (1-p_c)p_{AA}\Big)^{E(\widetilde{N}_r^{HAACK})}. \tag{4.68}$$

$p_c + (1-p_c)p_{AA}$ ist die Wahrscheinlichkeit, dass keine aggregierte Quittung gesendet werden kann oder dass die aggregierte Quittung verloren geht. Nun kann die Gesamtanzahl der Unicast-Quittungsnachforderungen aus der Anzahl der Quittungsnachforderungsrunden berechnet werden:

$$E(L_{aaq}^{HAACK}) = \sum_{k=1}^{E(O_q^{HAACK})} E(B_{aa})\hat{p}^{(k-1)}. \tag{4.69}$$

In der ersten Runde müssen an alle $B_{aa}$ Empfänger Quittungsnachforderungen gesendet werden, folglich ist $\hat{p}^{(k-1)}$ für $k=1$ ebenfalls eins. In jeder weiteren Runde fallen hingegen weniger Quittungsnachforderungen an, was durch $\hat{p}^{(k-1)}$ berücksichtigt wird. Die Anzahl der empfangenen Quittungsnachforderungen bei einem Empfänger ist:

$$E(\widetilde{L}_{aaq}^{HAACK}) = \frac{1}{E(B_{aa})} E(L_{aaq}^{HAACK})(1-p_q), \tag{4.70}$$

wobei $1/B_{aa}$ die Wahrscheinlichkeit ist, ein Empfänger zu sein, der eine Quittungsnachforderung erhält und $p_q$ die Wahrscheinlichkeit für Verlust der Quittungsnachforderungen ist. Die Anzahl empfangener aggregierter Quittungen bei einem beliebigen Gruppenführer, $\widetilde{L}_{aa}^{HAACK}$, setzt sich zusammen aus den aggregierten Quittungen aus der Übertragungswiederholungsphase und denen aus der Quittungsnachforderungsphase. Die aggregierten Quittungen aus der Quittungsnachforderungsphase sind genau eine Quittung von jedem noch fehlenden Empfänger $B_{aa}$:

$$E(\widetilde{L}_{aa}^{HAACK}) = BE(\widetilde{N}_r^{HAACK})(1-p_{AA})(1-p_c) + E(B_{aa}). \tag{4.71}$$

#### 4.5.1.7 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung und aggregierten Quittungen (K-HANAPP)

$E(M^{HANAPP})$ und $E(\widetilde{L}^{HANAPP})$ sind identisch zur Protokollklasse K-HNAPP. Unter Zuhilfenahme der Formeln 4.66, 4.67 und 4.69 lässt sich die Anzahl der Quittungsnachforderungen, $L_{aaq}^{HANAPP}$, bestimmen. Bei der Berechnung muss allerdings berücksichtigt werden, dass die Anzahl der abzufragenden Mitglieder der lokalen Gruppe, $B_{aa}$, aufgrund des differierenden Protokollverhaltens neu bestimmt werden muss. Jeder Empfänger sendet eine aggregierte Quittung selbständig nach einer bestimmten Anzahl erfolgreich empfangener Datennachrichten. $B_{aa}$ ist damit äquivalent zur Anzahl der verlorenen aggregierten Quittungen:

$$E(B_{aa}) \quad = \quad Bp_{AA}. \tag{4.72}$$

Da die fehlenden aggregierten Quittungen selektiv mit Unicast angefordert werden, erhält ein Gruppenführer am Ende von jedem Mitglied seiner lokalen Gruppe genau eine aggregierte Quittung:

$$E(\widetilde{L}_{aa}^{HANAPP}) \quad = \quad B. \tag{4.73}$$

$\widetilde{L}_{aaq}^{HANAPP}$, die Anzahl der empfangenen Nachforderungsnachrichten, folgt aus der Anzahl gesendeter Nachforderungsnachrichten $L_{aaq}^{HANAPP}$, der Nachrichtenverlustwahrscheinlichkeit und der Wahrscheinlichkeit einer der Empfänger zu sein, die eine Nachforderungsnachricht erhalten entsprechend der Formel 4.70. $L^{HANAPP}$, $R_N$, $O^{HANAPP}$, $O_r^{HANAPP}$ und $L_\phi^{HANAPP}$ können analog zur Protokollklasse K-NAPP und K-HNAPP berechnet werden.

### 4.5.2 Analyse der Bandbreite

Die Analyse des Bandbreitenbedarfs erfolgt aufgrund einer Betrachtung der Endpunkte von Kommunikationskanälen auf Transportschicht, d.h. des Senders und der Empfänger. Auf die Definition eines Netzwerkmodells zur Bandbreitenanalyse im Innern wurde aus guten Gründen verzichtet (siehe Abschnitt 4.4). Die hier vorgestellten Analysen sollen auf beliebige Netzwerkstrukturen anwendbar sein. Zudem ist der Flaschenhals der Übertragung oftmals gerade an den Kommunikationsendpunkten vorhanden. Bereits ein langsamer Empfänger, der beispielsweise über eine schlechte Modem-Verbindung angebunden ist, bremst die gesamte Übertragung aus. Schließlich ist bei volumenbasierter Abrechnung der Bandbreitenbedarf an den Kommunikationsendpunkten für die anfallenden Kosten verantwortlich. Aus diesen Gründen wird der Bandbreitenbedarf der Kommunikationskanäle auf Transportschicht bestimmt.

Im Folgenden werden die Analysen für die unterschiedlichen Protokollklassen dargelegt. Diese wurden erstmals in [Maihöfer 2000a, Maihöfer & Rothermel 2001a] vorgestellt. Die Bandbreitenanalyse

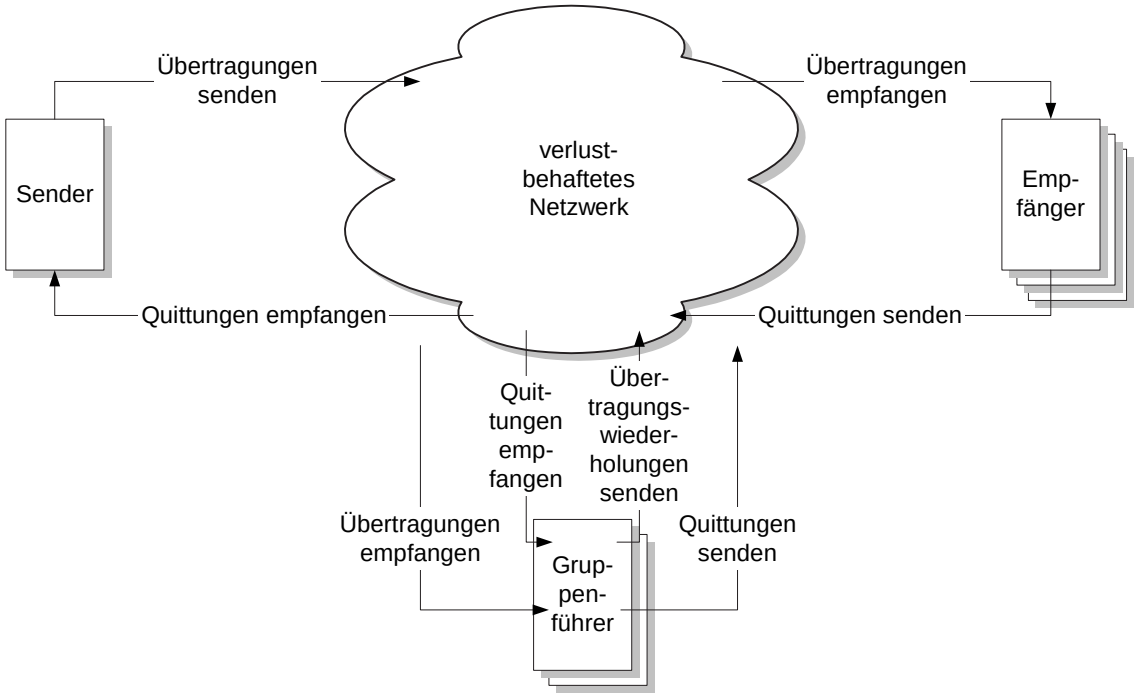



*Abbildung 4.5: Bandbreitenbedarf eines hierarchischen Protokolls*

der letzten beiden Protokollklassen mit aggregierten Quittungen, K-HAACK und K-HANAPP, folgt unmittelbar aus den Berechnungen aus Abschnitt 4.5.1 und der Bandbreitenanalyse aller anderen Klassen. Daher ist deren Analyse im Anhang wiedergegeben. Für die nun folgenden Berechnungen gibt Abbildung 4.5 einen Überblick über die zu berücksichtigenden Nachrichten eines hierarchischen Protokolls.

#### 4.5.2.1 Senderinitiiertes Protokoll (K-ACK)

Der Bandbreitenbedarf des Senders, $W_S^{ACK}$, und eines beliebigen Empfängers, $W_R^{ACK}$, wird auf der Grundlage einer einzelnen Datennachricht bestimmt, die erfolgreich an alle Empfänger gesendet werden soll. Der Sender sendet die Datennachricht und wartet anschließend auf alle positiven Quittungen. Auf fehlende Bestätigungen reagiert der Sender mit einer Übertragungswiederholung. Angenommen wird, dass vor dem Senden der Übertragungswiederholung alle nicht verlorenen Quittungen beim Sender eingegangen sind, d.h. dass der Sender keine verfrühte Entscheidung über die Notwendigkeit einer Übertragungswiederholung trifft.[1] Um den Bandbreitenbedarf zu bestimmen, müssen alle aus– und eingehenden Nachrichten berücksichtigt werden. Der Sender sendet Datennachrichten und

[1] Reale Protokollimplementierungen messen deshalb die Verzögerung zwischen Sender und Empfängern, um unnötige Übertragungswiederholungen zu vermeiden (siehe beispielsweise [Paul et al. 1997].)

empfängt Quittungen. Der Bandbreitenbedarf beim Sender kann folgendermaßen bestimmt werden:

$$\begin{aligned} W_S^{ACK} &= \textit{(initiale Übertragung)} + \textit{(Übertragungswiederholungen)} + \textit{(empfangene Quittungen)} \\ W_S^{ACK} &= W_d(1) + \sum_{m=2}^{M^{ACK}} W_d(m) + \sum_{i=1}^{\widetilde{L}^{ACK}} W_a(i). \end{aligned} \tag{4.74}$$

$W_d(m)$ und $W_a(i)$ ist dabei der notwendige Bandbreitenbedarf für die m-te bzw. i-te Übertragung einer Daten– oder Quittungsnachricht. Die Gesamtanzahl der notwendigen Übertragungswiederholungen $M^{ACK}$ und die Gesamtanzahl der positiven Quittungen $\widetilde{L}^{ACK}$ wurden bereits in Abschnitt 4.5.1.1 bestimmt. Mit Hilfe der Erwartungswerte kann die obige Formel vereinfacht werden zu:

$$E(W_S^{ACK}) = E(M^{ACK})E(W_d) + E(\widetilde{L}^{ACK})E(W_a). \tag{4.75}$$

Für die Bestimmung des Bandbreitenbedarfs bei einem beliebigen Empfänger wird berücksichtigt, dass gesendete Datennachrichten mit einer Wahrscheinlichkeit $(1-q_D)$ verloren gehen und dass jede ankommende Datennachricht mit einer positiven Quittung bestätigt wird:

$$W_R^{ACK} = \textit{(empfangene Übertragungen)} + \textit{(gesendete positive Quittungen)} \tag{4.76}$$

$$E(W_R^{ACK}) = E(M^{ACK})(1-q_D)\Big(E(W_d) + E(W_a)\Big). \tag{4.77}$$

Aus dem ermittelten Bandbreitenbedarf für Sender und Empfänger erfolgt die Berechnung zweier wichtiger Größen. Zum einen kann daraus der gesamte Bandbreitenbedarf auf Transportschicht für die Multicast-Gruppe berechnet werden. Dieser kann als die Kosten einer Multicast-Übertragung an den Kommunikationsendpunkten angesehen werden. Zum anderen ist aber auch der maximale Durchsatz auf der Transportschicht ermittelbar, der mit einer bestimmten Protokollklasse erzielt werden kann, unter der Annahme, dass im Netzwerk die Kapazität nicht begrenzt ist.

Für die Bestimmung des gesamten Bandbreitenbedarfs auf Transportschicht ist der Bedarf des Senders mit allen $R$ Empfängern zu addieren:

$$E(W^{ACK}) = E(W_S^{ACK}) + RE(W_R^{ACK}). \tag{4.78}$$

Der maximale durch den Bandbreitenbedarf des Senders limitierte Durchsatz ist der Kehrwert des Bandbreitenbedarfs:

$$\Lambda_S^{ACK} = \frac{1}{E(W_S^{ACK})}. \tag{4.79}$$

Eine alternative Bezeichnung von $\Lambda$ könnte auch Bandbreiteneffizienz lauten. Je größer der Wert von $\Lambda$, desto besser wird die verfügbare Bandbreite nutzbringend eingesetzt. Entsprechend lautet für den Empfänger der maximal zu erzielende Durchsatz, $\Lambda_R^{ACK}$:

$$\Lambda_R^{ACK} = \frac{1}{E(W_R^{ACK})}. \tag{4.80}$$

Der Gesamtdurchsatz, der mit einem nicht hierarchischen Protokoll erzielt werden kann, ist der Minimalwert des Durchsatzes von Sender und Empfänger. Mit anderen Worten, die Komponente mit dem höchsten Bandbreitenbedarf limitiert den Gesamtdurchsatz. Der Gesamtdurchsatz ist gegeben durch:

$$\Lambda^{ACK} = min\{\Lambda_S^{ACK}, \Lambda_R^{ACK}\}. \tag{4.81}$$

#### 4.5.2.2 Empfängerinitiiertes Protokoll (K-NACK)

Auch bei den empfängerinitiierten Protokollen wird die Annahme getroffen, dass alle negativen Quittungen vom Sender bereits empfangen sind, bevor es zur Einleitung einer Übertragungswiederholung kommt. Der Bandbreitenbedarf des Senders setzt sich zusammen aus den Übertragungen und Quittungen:

$$\begin{aligned} W_S^{NACK} &= \textit{(gesendete Übertragungen)} + \textit{(empfangene Quittungen)} \\ W_S^{NACK} &= \sum_{m=1}^{M^{NACK}} W_d(m) + \sum_{i=1}^{\widetilde{L}^{NACK}} W_n(i) \qquad (4.82) \\ E(W_S^{NACK}) &= E(M^{NACK})E(W_d) + E(\widetilde{L}^{NACK})E(W_n). \qquad (4.83) \end{aligned}$$

Der Bandbreitenbedarf des Empfängers ist:

$$E(W_R^{NACK}) = E(M^{NACK})(1-q_D)E(W_d) + \Big(E(O_r^{NACK}) - 1\Big)E(W_n). \tag{4.84}$$

Die Anzahl der Runden für einen beliebigen Empfänger $O_r^{NACK}$ ist gleich oder größer als die Anzahl der Übertragungen, da durch den Verlust von negativen Quittungen zusätzliche Quittungen notwendig sind. Die Berechnung ist der Formel 4.25 zu entnehmen. Zu beachten gilt, dass die letzte, erfolgreiche Übertragung keine Quittungen auslöst, weswegen die Rundenanzahl um eins erniedrigt wird. Der Durchsatz und der gesamte Bandbreitenbedarf lässt sich analog zur Protokollklasse K-ACK bestimmen.

#### 4.5.2.3 Empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-NAPP)

Die Berechnung des Bandbreitenbedarfs beim Sender erfolgt analog zur vorigen Protokollklasse:

$$E(W_S^{NAPP}) = E(M^{NAPP})E(W_d) + E(\widetilde{L}^{NAPP})E(W_n). \tag{4.85}$$

Der Bandbreitenbedarf des Empfängers kann nun bestimmt werden zu:

$$\begin{aligned} E(W_R^{NAPP}) &= E(M^{NAPP})(1-q_D)E(W_d) + \underbrace{\Big(E(O_r^{NAPP}) - 1\Big)\frac{E(L_\phi^{NAPP})}{E(R_N)}E(W_n)}_{\textit{gesendete Quittungen}} \\ &\quad + \underbrace{\Big[\Big(E(O^{NAPP}) - 1\Big)E(L_\phi^{NAPP}) - \Big(E(O_r^{NAPP}) - 1\Big)\frac{E(L_\phi^{NAPP})}{E(R_N)}\Big](1-q_N)E(W_n)}_{\textit{empfangene Quittungen}}. \qquad (4.86) \end{aligned}$$

Für die Berechnung der fehlenden Größen siehe Abschnitt 4.5.1.3. Wie der Formel zu entnehmen ist, werden nicht nur Quittungen gesendet, sondern ein Empfänger empfängt auch Quittungen anderer Empfänger, die per Multicast gesendet wurden. Der Term $E(L_{\phi}^{NAPP})/E(R_N)$ bestimmt die durchschnittliche Anzahl gesendeter Quittungen eines Empfängers pro Runde. Die empfangenen Quittungen setzen sich zusammen aus den gesendeten Quittungen aller Empfänger abzüglich der eigenen gesendeten Quittungen. Der Durchsatz und der gesamte Bandbreitenbedarf lässt sich analog zu den vorhergehenden Protokollklassen bestimmen.

#### 4.5.2.4 Hierarchisches senderinitiiertes Protokoll (K-HACK)

Bei den hierarchischen Protokollen ist eine Unterscheidung zwischen dem Bandbreitenbedarf des Senders, $W_S^{HACK}$, allen weiteren Gruppenführern, $W_G^{HACK}$, und den Empfängern die keine Gruppenführer sind, $W_R^{HACK}$, sinnvoll. In der Analyse wird die Annahme getroffen, dass alle lokalen Gruppen $B$ Mitglieder besitzen, wie es mittels TRS-M erreicht werden kann.

**Sender:** Die Berechnung des Bandbreitenbedarfs des Senders ist äquivalent zu den Berechnungen für die Protokollklasse K-ACK. Unterschiedliche Resultate ergeben sich allerdings durch unterschiedliche Werte für $M^{HACK}$ und $\widetilde{L}^{HACK}$ (siehe Formeln 4.45 und 4.46). Der Bandbreitenbedarf beträgt:

$$E(W_S^{HACK}) = E(M^{HACK})E(W_d) + E(\widetilde{L}^{HACK})E(W_a). \tag{4.87}$$

**Empfänger (ohne Gruppenführer):** Als Empfänger sollen hier reine Empfänger ohne Gruppenführerrolle verstanden werden. Berücksichtigung findet die Anzahl an Übertragungen des Vaterknotens im Kontrollbaum, $\widetilde{N}_r^{HACK}$, auf die mit einer positiven Quittung reagiert wird und alle Übertragungen auch anderer lokaler Gruppen, $\widetilde{N}_{r,t}^{HACK}$. Der gesamte Bandbreitenbedarf eines Empfängers ist demnach:

$$E(W_R^{HACK}) = E(\widetilde{N}_{r,t}^{HACK})E(W_d) + E(\widetilde{N}_r^{HACK})E(W_a). \tag{4.88}$$

**Gruppenführer (ohne Sender):** Obwohl der Sender ebenfalls ein Gruppenführer ist, gilt für ihn der bereits dargelegte Bandbreitenbedarf, der von dem der anderen Gruppenführer abweicht. In den folgenden Berechnungen sollen grundsätzlich alle Gruppenführer ohne den Sender betrachtet werden, jedoch ohne dies jedesmal explizit zu erwähnen. Grundsätzlich ist ein Gruppenführer sowohl ein Sender als auch ein Empfänger. Folglich ist sein Bandbreitenbedarf die Summe des Bandbreitenbedarfs von Sender und Empfänger. Zwei Abweichungen müssen allerdings Berücksichtigung finden. Erstens sendet ein Gruppenführer nicht die initiale Übertragung, die nur der Sender vornimmt und zweitens empfängt ein Gruppenführer nicht wie ein Empfänger von $G-1$ weiteren lokalen Gruppen Übertragungen, sondern nur von $G-2$, da er selbst Mitglied von zwei lokalen Gruppen ist. Dies ist in

der Berechnung von $E(\widetilde{N}_g^{HACK})$ (siehe Formel 4.49) bereits berücksichtigt. Der Bandbreitenbedarf ist insgesamt:

$$\begin{aligned} E(W_G^{HACK}) &= \Big(E(M^{HACK})-1\Big)E(W_d)+E(\widetilde{L}^{HACK})E(W_a)+E(\widetilde{N}_g^{HACK})E(W_d)+E(\widetilde{N}_r^{HACK})E(W_a) \\ &= E(W_S^{HACK})+E(W_R^{HACK})-E(W_d(1))-\Big(E(M^{HACK})-1\Big)(1-q_D)p_l E(W_d). \end{aligned} \quad (4.89)$$

Die maximalen Durchsatzraten $\Lambda_S^{HACK}$, $\Lambda_R^{HACK}$, $\Lambda_G^{HACK}$ für den Sender, Empfänger und Gruppenführer sind:

$$\Lambda_S^{HACK}=\frac{1}{E(W_S^{HACK})}\ ,\ \Lambda_R^{HACK}=\frac{1}{E(W_R^{HACK})}\ ,\ \Lambda_G^{HACK}=\frac{1}{E(W_G^{HACK})}. \quad (4.90)$$

Der Gesamtdurchsatz, $\Lambda^{HACK}$, der mit einem hierarchischen Protokoll erzielt werden kann, ist der Minimalwert des Durchsatzes von Sender, Empfänger und Gruppenführer:

$$\Lambda^{HACK}=min\{\Lambda_S^{HACK},\Lambda_G^{HACK},\Lambda_R^{HACK}\}. \quad (4.91)$$

Der gesamte Bandbreitenbedarf ist die Summe des Bandbreitenbedarfs des Senders, aller Empfänger und aller Gruppenführer:

$$W^{HACK} = W_S^{HACK}+(R-G+1)W_R^{HACK}+(G-1)W_G^{HACK}. \quad (4.92)$$

#### 4.5.2.5 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-HNAPP)

Die Klasse K-HNAPP erweitert die Klasse K-NAPP nicht nur um den Kontrollbaum, sondern auch um die periodischen positiven Quittungen. Die numerischen Auswertungen berücksichtigen, dass periodische Quittungen einen wesentlich geringeren Aufwand zum Senden und Empfangen verursachen, da sie nicht nach jeder Datenübertragung gesendet werden. Dazu ist der Bandbreitenbedarf für eine periodische Quittung $W_\Phi$ entsprechend klein zu wählen. Beispielsweise kann ein Zehntel des Wertes von $W_n$ angenommen werden, vorausgesetzt, periodische Quittungen werden nach jeweils zehn empfangenen Datennachrichten versendet.

**Sender:** Zusätzlich zu den Datennachrichten und den Quittungen müssen die periodischen Quittungen berücksichtigt werden. Außerdem sind negative Quittungen anderer lokaler Gruppen zu empfangen, die bereits in $E(\widetilde{L}^{HNAPP})$ (siehe Formel 4.53) berücksichtigt sind:

$$E(W_S^{HNAPP}) = E(M^{HNAPP})E(W_d)+E(\widetilde{L}^{HNAPP})E(W_n)+E(S)E(W_\Phi). \quad (4.93)$$

**Empfänger (ohne Gruppenführer):** Der Empfänger kann sowohl Übertragungswiederholungen als auch negative Quittungen empfangen, die für eine andere lokale Gruppe bestimmt sind. Diese müssen

zwar empfangen werden, ändern aber nichts am Protokollverhalten, da fremde Übertragungswiederholungen nicht bestätigt werden und fremde Quittungen nicht zur Quittungsunterdrückung führen. Der Bandbreitenbedarf für einen Empfänger ist:

$$
\begin{aligned}
E(W_R^{HNAPP}) &= E(M^{HNAPP})(1-q_D)E(W_d) + E(W_\Phi) \\
&+ \Big(E(O_r^{HNAPP})-1\Big)\frac{E(L_\phi^{HNAPP})}{E(R_N)}E(W_n) \\
&\underbrace{+\Big[\Big(E(O^{HNAPP})-1\Big)E(L_\phi^{HNAPP}) - \Big(E(O_r^{HNAPP})-1\Big)\frac{E(L_\phi^{HNAPP})}{\mathrm{E}(R_N)}\Big](1-q_N)E(W_n)}_{\textit{eigene lokale Gruppe}} \\
&\underbrace{+(G-1)p_l\Big[\Big(E(M^{HNAPP})-1\Big)(1-q_D)E(W_d) + \Big(E(O^{HNAPP})-1\Big)E(L_\phi^{HNAPP})(1-q_N)E(W_n)\Big]}_{\textit{fremde lokale Gruppen}}.
\end{aligned}
\tag{4.94}
$$

$L_\phi^{HNAPP}$ wird entsprechend der Formel 4.43 bestimmt und $R_N$ kann analog zur Formel 4.44 der Protokollklasse K-NAPP bestimmt werden mit $B$ anstatt $R$ für die Empfänger einer Übertragungswiederholung. Mit den Erklärungen zur Protokollklasse K-NAPP lässt sich der Gedankengang der Berechnung erschließen.

**Gruppenführer (ohne Sender):** Analog zur vorigen Protokollklasse erfolgt auch hier die Berechnung des Bandbreitenbedarfs eines Gruppenführers zu:

$$
\begin{aligned}
E(W_G^{HNAPP}) &= E(W_S^{HNAPP}) + E(W_R^{HNAPP}) - E(W_d(1)) \\
&- p_l\Big[\Big(E(M^{HNAPP})-1\Big)(1-q_D)E(W_d) + \Big(E(O^{HNAPP})-1\Big)E(L_\phi^{HNAPP})(1-q_N))E(W_n)\Big].
\end{aligned}
\tag{4.95}
$$

Die zweite Zeile ist der Bandbreitenbedarf für das Empfangen von Nachrichten einer einzelnen fremden lokalen Gruppe, der subtrahiert werden muss. Der Gesamtdurchsatz und Gesamtbandbreitenbedarf wird äquivalent zur Protokollklasse K-HACK berechnet.

An dieser Stelle soll die Bandbreitenanalyse abgeschlossen werden, da die wesentlichen Konzepte der Berechnung vorgestellt wurden. Die noch verbleibenden zwei Protokollklassen, K-HAACK und K-HANAPP, sind im Anhang wiedergegeben.

### 4.5.3 Analyse der CPU-Belastung

In den vorigen Abschnitten wurde der Bandbreitenbedarf der verschiedenen Protokollklassen analysiert. Ähnlich zum Bandbreitenbedarf für eingehende und ausgehende Nachrichten wird auch CPU-Rechenzeit für jede Nachricht verbraucht. Zusätzlich werden Zeitgeber bei der CPU-Belastung berücksichtigt, da jeder Zeitgeberablauf zu einer Unterbrechungsbehandlung führt. Schließlich muss

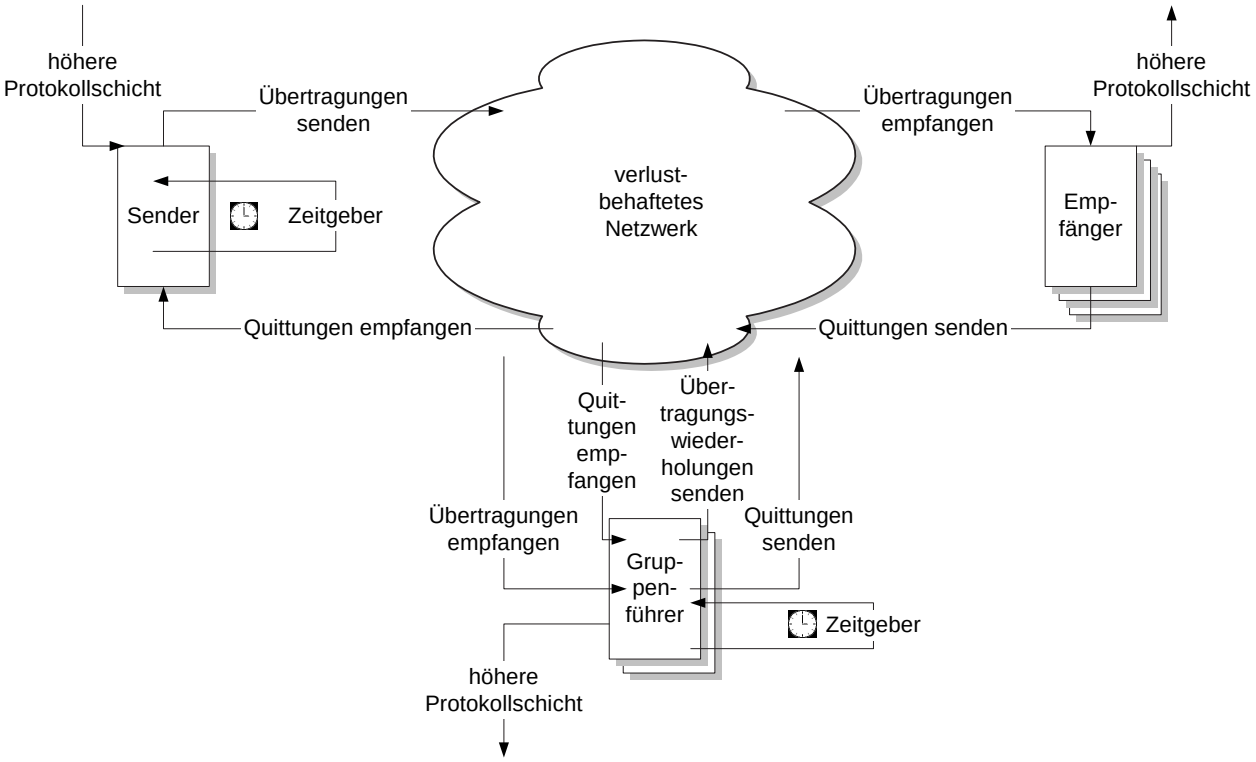



*Abbildung 4.6: CPU-Belastung eines senderinitiierten hierarchischen Protokolls*

mit der Übergabe der Daten von und an eine höhere Protokollschicht auch deren Bearbeitungsaufwand Berücksichtigung finden. In Abbildung 4.6 sind die zu berücksichtigenden Komponenten für ein senderinitiiertes hierarchisches Verfahren dargestellt. Das Ergebnis dieser Analyse ist wiederum der erzielbare maximale Durchsatz einer Protokollklasse. Die Analysen zur CPU-Belastung wurden erstmals in [Maihöfer et al. 2000, Maihöfer & Rothermel 2001f] vorgestellt.

Der folgende Abschnitt zeigt beispielhaft die Berechnungsschritte zur Ermittlung der CPU-Belastung für die flache Protokollklasse K-ACK. Dabei wird trotz der zusätzlichen Komponenten auch ersichtlich, dass die Vorgehensweise sehr ähnlich zur Bandbreitenanalyse ist. Aus diesem Grund sind die restlichen Protokollklassen hier nicht weiter analysiert. Im Anhang findet sich deren komplette Berechnung.

#### 4.5.3.1 Senderinitiiertes Protokoll (K-ACK)

Die CPU-Belastung des Senders wird mit $C_S^{ACK}$ und die des Empfängers mit $C_R^{ACK}$ bezeichnet. Eine Datenübertragung beginnt mit der Übergabe von Daten einer höheren Protokollschicht und dem Zusammensetzen der eigentlichen Datennachricht. Zusätzlich muss ein Zeitgeber (engl. Timer) initiiert werden. Eine Datenübertragung verursacht den Aufwand $X_d$. Nachdem eine Datenübertragung

erfolgt ist, werden alle positiven Quittungen vom Sender gesammelt. Der Aufwand für das Empfangen einer positiven Quittung ist $X_a$. Sind von allen Empfängern positive Quittungen eingetroffen, so war die Übertragung erfolgreich. Ansonsten wird nach einer bestimmten Zeitvorgabe eine Übertragungswiederholung ausgelöst und eine neue Zeitvorgabe gesetzt. Das Auslösen und Setzen einer neuen Zeitvorgabe verursacht den Aufwand $X_t$. Alle Datenübertragungen mit Ausnahme der letzten, erfolgreichen, bedingen einen neuen Zeitgeber zu initiieren. Der gesamte Bearbeitungsaufwand beim Sender ist somit:

$$\begin{aligned} C_S^{ACK} &= \textit{(höhere Protokollschicht)} + \textit{(initiale Übertragung)} \\ &\quad + \textit{(Zeitgeber)} + \textit{(Übertragungswiederholungen)} + \textit{(empfangene Quittungen)} \\ C_S^{ACK} &= X_f + X_d(1) + \sum_{m=2}^{M^{ACK}} \Big( X_t(m) + X_d(m) \Big) + \sum_{i=1}^{\widetilde{L}^{ACK}} X_a(i) \qquad (4.96) \\ E(C_S^{ACK}) &= E(X_f) + E(M^{ACK})E(X_d) + \Big( E(M^{ACK}) - 1 \Big) E(X_t) + E(\widetilde{L}^{ACK})E(X_a). \qquad (4.97) \end{aligned}$$

$M^{ACK}$ und $\widetilde{L}^{ACK}$ wurden in Abschnitt 4.5.1.1 bestimmt. Der maximale Durchsatz aus Sicht des Senders ergibt sich aus:

$$\Lambda_S^{ACK} = \frac{1}{E(C_S^{ACK})}. \qquad (4.98)$$

Für den Empfänger wird das Senden und Empfangen von Nachrichten mit $Y$ statt mit $X$ bezeichnet. Da ein Empfänger eines senderinitiierten Protokolls keine Zeitgeber verwalten muss, ist die Bestimmung der CPU-Belastung ähnlich der Bestimmung des Bandbreitenbedarfs:

$$\begin{aligned} C_R^{ACK} &= \textit{(Übertragungen empfangen)} + \textit{(Quittungen senden)} + \textit{(höhere Protokollschicht)} \\ E(C_R^{ACK}) &= E(M^{ACK})(1 - q_D) \Big( E(Y_d) + E(Y_a) \Big) + E(Y_f). \qquad (4.99) \end{aligned}$$

Die maximale Durchsatzrate für einen Empfänger, $\Lambda_R^{ACK}$, ist:

$$\Lambda_R^{ACK} = \frac{1}{E(C_R^{ACK})}. \qquad (4.100)$$

Der gesamte Systemdurchsatz, $\Lambda^{ACK}$, ergibt sich aus dem Minimum von Sender– und Empfängerdurchsatz:

$$\Lambda^{ACK} = min\{\Lambda_S^{ACK}, \Lambda_R^{ACK}\}. \qquad (4.101)$$

### 4.5.4 Analyse der Verzögerung

Die Analyse der Verzögerung berücksichtigt drei Verzögerungswerte. Die durchschnittliche Auslieferungsverzögerung ist die Zeitspanne zwischen dem initialen Eintreffen oder der Generierung einer Datennachricht beim Sender und dem erfolgreichen Empfang bei einem zufällig gewählten Empfänger. Die Auslieferungsverzögerung ist vor allem für Echtzeitanwendungen von großer Bedeutung. Beispiele dafür sind verteilte interaktive Simulationen, verteilte Spiele aber auch kodierte Videoübertragungen. Hingegen ist die Umlaufverzögerung die durchschnittliche Zeit zwischen dem initialen Eintreffen oder Generierung einer Datennachricht beim Sender und dem Empfang aller dazugehörigen Quittungen. Die Umlaufverzögerung ist für den Sender der Daten und den Durchsatz von Interesse. Nachdem eine ausgelieferte Nachricht von allen Empfängern bestätigt wurde, kann der Sender diese löschen und den damit belegten Speicherplatz wieder freigeben. Verwendet das Protokoll einen fensterbasierten Sendemechanismus, hat die Umlaufverzögerung direkten Einfluss auf den maximal erzielbaren Durchsatz. Der Zusammenhang ist Durchsatz = Fenstergröße / Umlaufverzögerung [Allman et al. 1999]. Je kleiner die Umlaufverzögerung, desto schneller wird das Sendefenster weitergeschaltet und desto höher ist der erreichbare Durchsatz.

Neben der durchschnittlichen Auslieferungsverzögerung und der durchschnittlichen Umlaufverzögerung, wird die maximale Auslieferungsverzögerung betrachtet, bis alle Empfänger eine Datennachricht korrekt empfangen haben. In bisherigen Arbeiten über Verzögerungsanalysen wurde lediglich die durchschnittliche Auslieferungsverzögerung, nicht aber die maximale Auslieferungsverzögerung betrachtet. Bei hinreichend kleiner Nachrichtenverlustwahrscheinlichkeit liegt die durchschnittliche Auslieferungsverzögerung nur wenige Prozente über der reinen Übertragungszeit im Netzwerk, da vergleichsweise selten Übertragungswiederholungen stattfinden. Dadurch wird zwischen den verschiedenen Protokollklassen kein wesentlicher Unterschied in der Leistung ihrer Fehlerbehebungsalgorithmen festgestellt. Die Betrachtung der Maximalverzögerung liefert wesentlich genauere Anhaltspunkte über die Leistung der Fehlerbehebungsalgorithmen. Ein weiterer Vorteil ergibt sich aus der Kenntnis der Maximalverzögerung für der Praxis. Sind Anwendungen auf garantierte Zeitschranken für die Auslieferungsverzögerung angewiesen, kann direkt auf die Ergebnisse für die Maximalverzögerung zurückgegriffen werden. Die Wahrscheinlichkeit für den korrekten Empfang bei allen Empfängern ist in der Analyse wählbar zwischen $p_D \leq p < 1$, je nachdem mit welcher Wahrscheinlichkeit eine Anwendung korrekten Empfang wünscht. Beispielsweise kann eine Anwendung damit garantieren, dass innerhalb einer bestimmten Zeitschranke im Mittel bei $p \cdot 100\%$ der Empfänger die Nachricht empfangen wurde. Neben zeitkritischen Anwendungen, die sich für die Maximalverzögerung interessieren, ist diese in der Praxis zudem für die Einstellungen der Zeitgeber für Übertragungswiederholungen und Fenstergrößen für die Flusskontrolle bei der Protokollkonfiguration von großem Interesse. Die drei analysierten Verzögerungszeiten sind in Abbildung 4.7 dargestellt.

Die hier dargestellte Analyse wurde für flache Protokollklassen erstmals in [Maihöfer & Rothermel

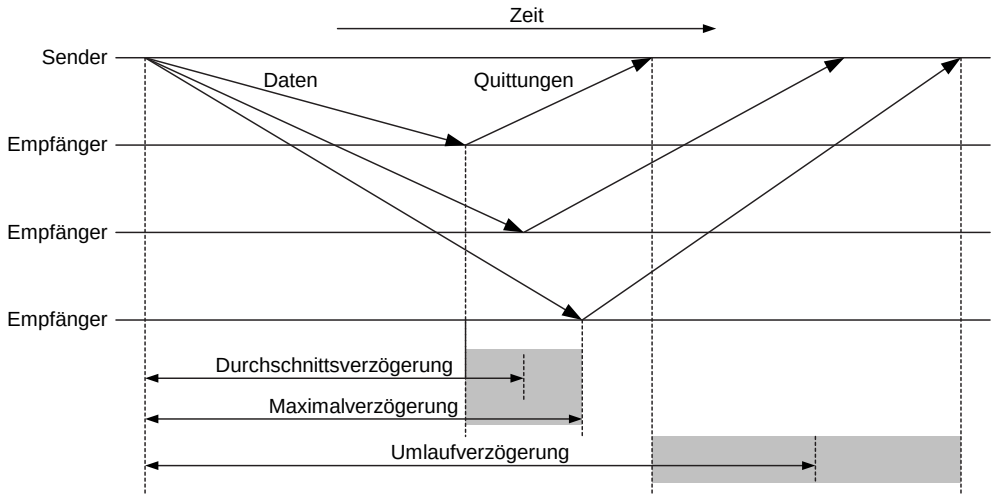



*Abbildung 4.7: Verzögerungs– und Umlaufzeiten*

2001c] und für hierarchische Protokollklassen in [Maihöfer & Rothermel 2001b, Maihöfer & Rothermel 2001d] vorgestellt. Schließlich wurde in [Maihöfer & Rothermel 2001e] eine Verzögerungsanalyse, aufbauend auf räumlich abhängige Nachrichtenverluste, durchgeführt (vgl. Abschnitte 4.4 und 4.5.5).

Im Folgenden werden die Grundlagen für das Verständnis der Verzögerungsanalyse vermittelt. Die Verzögerung einer Nachricht wird neben der reinen Übertragungszeit im Netzwerk, diese wird mit $\tau$ bezeichnet, durch die Warte– und Bearbeitungszeiten auf den Sende– und Empfangssystemen bestimmt. Die wesentliche Größe ist wiederum die Anzahl der notwendigen Übertragungen bis zum korrekten Empfang einer Datennachricht bei allen Empfänger, wie sie bereits in Abschnitt 4.5.1 bestimmt wurde.

Die Bearbeitungszeit wird durch ein Warteschlangensystem ermittelt. Beispiele für Warteschlangen sind Personen vor den Kassen eines Supermarkts oder Prozesse, die auf die Bearbeitung durch einen oder mehrere Prozessoren warten. Allgemein spricht man in der Warteschlangentheorie von Forderungen, die an einer Bearbeitungseinheit eintreffen. Die Zeit zwischen dem Eintreffen von Forderungen als auch die Länge der Bearbeitungszeit einer Forderung sind stochastische Größen. Bei einer Vielzahl von Warteschlangenmodellen gilt die Annahme eines poissonverteilten Forderungseingangs als statistisch gesichert [Kleinrock 1975, Rüegg 1986]. Die Poissonverteilung ist gegeben durch [Rüegg 1986]:

$$P(N = n) \quad = \quad \frac{\lambda^n}{n!} e^{-\lambda}, \tag{4.102}$$

mit $n$ als der Anzahl von Forderungen bis zu einem bestimmten Zeitpunkt und $\lambda$ als einen wählbaren Parameter, der den Erwartungswert bestimmt. Die Poissonverteilung resultiert in exponentiell verteilten Zwischenankunftszeiten [Kleinrock 1976, S. 65]. Die Exponentialverteilung hat die Eigenschaft

der Gedächtnislosigkeit. Diese besagt, dass die Vergangenheit einer Zufallsvariable keine Bedeutung für die Vorhersage der Zukunft besitzt. Folgendes Beispiel veranschaulicht dies. Angenommen, ein Telefonanruf erfolgte zum Zeitpunkt $t_0$. Wenn nun zum Zeitpunkt $t_0$ gefragt wird, wieviel Zeit vergehen wird bis zum nächsten Telefonanruf, dann liefert die Exponentialverteilung die probabilistische Antwort, beispielsweise durchschnittlich fünf Sekunden. Wird die selbe Frage zu einem späteren Zeitpunkt $t_1$ erneut gestellt, liefert die Exponentialverteilung erneut die selbe Antwort, nämlich durchschnittlich fünf Sekunden. Dieses Verhalten ist auch für Nachrichtenankunftszeiten in Kommunikationsnetzen realistisch [Kleinrock 1976].

Nun soll erneut das Ausgangsproblem betrachtet werden. Die Bearbeitungszeit für eine Nachricht auf einem Endsystem ist durch dessen Last, $\rho$, bestimmt. Die Last wiederum ist durch die Bearbeitung von Daten– und Quittungsnachrichten bestimmt. Infolgedessen wird die Anzahl von Daten– und Quittungsnachrichten bestimmt, die auf einem Endsystem bearbeitet werden müssen oder mit anderen Worten, die Raten (Nachrichten pro Zeiteinheit, $\lambda$) für ausgehende und eingehende Nachrichten. Ausgehend von den Nachrichtenraten wird die Bearbeitungszeit bestimmt.

In Warteschlangensystemen wird die Last auch Verkehrsintensität genannt. Allgemein ausgedrückt ist sie das Produkt der Verkehrsrate $\lambda$ und der mittleren Bearbeitungszeit für eine Anfrage $E(S)$ [Kleinrock 1976]:

$$\rho = \lambda E(S). \tag{4.103}$$

Die Verkehrsrate ist hier durch die Nachrichtenrate gegeben. Die Last eines Endsystems ist folglich die Summe aller Nachrichtenraten multipliziert mit deren Bearbeitungszeiten. Die Bearbeitungszeiten werden in den numerischen Auswertungen mit real gemessenen Zeiten belegt.

Ankunftszeiten von Anfragen werden in der Warteschlangentheorie üblicherweise mit der Poissonverteilung modelliert, wie oben bereits ausgeführt wurde. Für die Bearbeitungszeit einer Anfrage wird keine spezielle Verteilung angenommen. Diese kann somit beliebig sein. In der Warteschlangentheorie wird ein solches System nach der Notation von Kendall als $M|G|1$ Warteschlange bezeichnet [Kleinrock 1976]. $M$ steht für Zwischenankunftszeiten, die der Markoffeigenschaft genügen. Die Eigenschaft der Gedächtnislosigkeit wird als Markoffeigenschaft bezeichnet. Deshalb erfüllen die exponentiell verteilten Zwischenankunftszeiten diese Bedingung. $G$ steht für beliebig verteilte Bearbeitungszeiten und 1 steht für die Anzahl paralleler Bearbeitungseinheiten.

Mit Hilfe der Pollaczek-Chintchine Formel und der Formel von Little lässt sich die mittlere Wartezeit für eine Nachricht bis zum Start der Bearbeitung in einem $M|G|1$ System folgendermaßen ermitteln [Kleinrock 1976]:

$$E(W) = \frac{\lambda E(S^2)}{2(1-\rho)}, \tag{4.104}$$

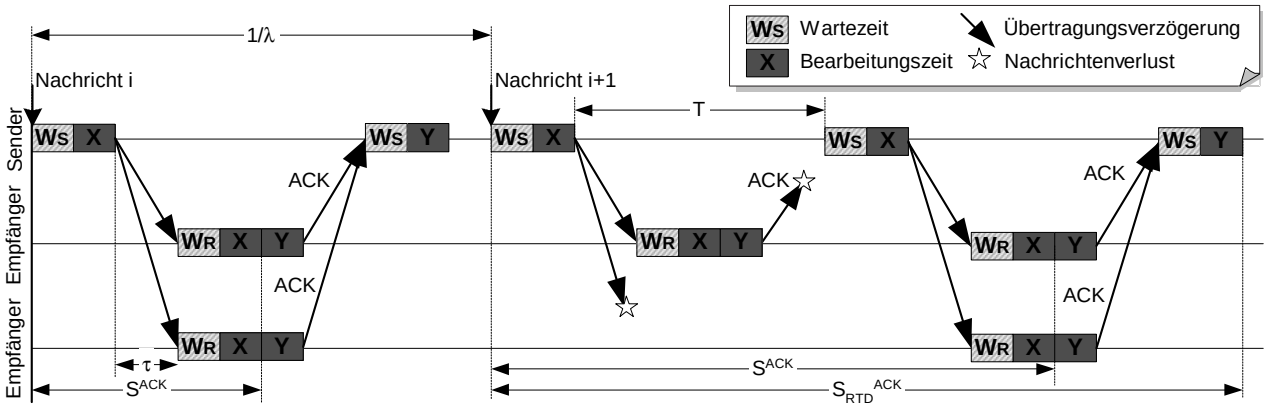



*Abbildung 4.8: Verzögerungskomponenten der Protokollklasse K-ACK*

wobei $S$ die Bearbeitungszeit für eine Forderung ist.

Damit wurden die wesentlichen Voraussetzungen für eine Verzögerungsanalyse erarbeitet. Mit der Formel 4.104 kann die Wartezeit, $W$, bis zur Bearbeitung einer Anfrage bestimmt werden. Dazu ist es notwendig, die Last, $\rho$, eines Endsystems zu bestimmen. Die Last ist basierend auf Formel 4.103 durch die Bearbeitungszeit und die Anfragerate, $\lambda$, bestimmt. Die Anfrageraten lassen sich aufgrund der Überlegungen aus Abschnitt 4.5.1 bestimmen. Nachdem die durchschnittliche Wartezeit einer Nachricht berechnet wurde, kann eine einzelne Nachricht betrachtet werden, die vom Sender zu einem beliebigen Empfänger übertragen wird. Das Ergebnis ist die durchschnittliche Auslieferungsverzögerung, die im Folgenden kurz Durchschnittsverzögerung genannt wird. Die Vorgehensweise für die Maximalverzögerung und die Umlaufverzögerung ist ähnlich und wird in den nachfolgenden Abschnitten behandelt. Die erarbeiteten Grundlagen werden nun auf die konkreten Protokollklassen angewendet.

#### 4.5.4.1 Senderinitiiertes Protokoll (K-ACK)

Alle relevanten Verzögerungskomponenten für die Protokollklasse K-ACK sind in Abbildung 4.8 beispielhaft dargestellt.

**Mittlere Wartezeit einer Nachricht beim Sender:** Um die obigen Formeln anwenden zu können, muss die Last des Senders ermittelt werden. Damit ist es möglich, die mittlere Wartezeit zwischen der Ankunft der ersten Nachricht beim Sender und dem eigentlichen Senden zu bestimmen. Für die Lastbestimmung werden die drei Nachrichtenflüsse betrachtet, die der Sender zu bearbeiten hat:

1. Fluss der Datennachrichten, die zum ersten Mal übertragen werden. Dieser Nachrichtenfluss wird mit $\lambda_t^S$ bezeichnet und hat die parametrisierbare Rate $\lambda$. Die Bearbeitungszeit für eine Datennachricht sei $X$.

2. Fluss der Übertragungswiederholungen aufgrund von Datenverlust. Dieser Nachrichtenfluss wird mit $\lambda_r^S$ bezeichnet und hat die Rate $\lambda(E(M^{ACK})-1)$, da eine Datennachricht $E(M^{ACK})-1$ Wiederholungen erfährt. $E(M^{ACK})$ ist die Gesamtanzahl von Übertragungen aus Abschnitt 4.5.1, um eine Datennachricht bei allen Empfängern korrekt zu empfangen.

3. Der letzte Nachrichtenfluss $\lambda_a^S$ ergibt sich durch die ankommenden positiven Quittungen mit Rate $\lambda RE(M^{ACK})(1-q_D)(1-p_A)$. Jeder Empfänger $R$ sendet eine Quittung für jede Datenübertragung $E(M^{ACK})$, vorausgesetzt die Datenübertragung geht nicht verloren mit Wahrscheinlichkeit $1-q_D$. Schließlich muss die Quittungsnachricht erfolgreich beim Sender ankommen, was durch die Wahrscheinlichkeit $1-p_A$ bestimmt wird. Die Bearbeitungszeit für eine positive Quittung sei $Y$.

Die Last wird durch die Formel 4.103 bestimmt. Für den Sender ergibt sich somit die Last, $\rho_S^{ACK}$, als Summe aller Nachrichtenraten multipliziert mit deren Bearbeitungszeiten:

$$\rho_S^{ACK} = \lambda E(M^{ACK})E(X)+\lambda RE(M^{ACK})(1-q_D)(1-p_A)E(Y). \tag{4.105}$$

Aus der Last des Senders kann mit Hilfe der Formel 4.104 die mittlere Wartezeit bis zum Bearbeitungsbeginn für eine Nachricht bestimmt werden:

$$E(W_S^{ACK}) = \frac{(\lambda_t^S+\lambda_r^S)E(X^2)+\lambda_a^S E(Y^2)}{2(1-\rho_S^{ACK})}. \tag{4.106}$$

**Mittlere Wartezeit einer Nachricht beim Empfänger:** Der Empfänger hat nur einen Nachrichtenfluss zu bearbeiten, den der ankommenden Datennachrichten. Für das Senden der Quittungen ergibt sich kein eigener Nachrichtenfluss, da diese direkt im Anschluss an den Empfang der Datennachrichten gesendet werden. Der Nachrichtenfluss wird mit $\lambda_t^R$ bezeichnet und hat die Rate $\lambda E(M^{ACK})(1-q_D)$. Die Bearbeitungszeit für den Empfang einer Datennachricht und das Senden der Quittung ist $X+Y$. Zu beachten ist, dass $X$ und $Y$ unabhängige Zufallsvariablen sind. Die Last des Empfängers ist:

$$\rho_R^{ACK} = \lambda E(M^{ACK})(1-q_D)\Big(E(X)+E(Y)\Big). \tag{4.107}$$

Daraus lässt sich für den Empfänger die mittlere Wartezeit einer Nachricht bis zur Bearbeitung ableiten:

$$E(W_R^{ACK}) = \frac{\lambda_t^R E\left((X+Y)^2\right)}{2(1-\rho_R^{ACK})}. \tag{4.108}$$

Da $X$ und $Y$ unabhängige Zufallsvariablen sind und laut [Rüegg 1986] gilt $E(V+W)=E(V)+E(W)$, $Var(V)=E(V^2)-(E(V))^2$ und $Var(V+W)=Var(V)+Var(W)$ für unabhängige Zufallsvariablen $V$ und $W$, kann die obige Formel wie folgt umgeformt werden:

$$E(W_R^{ACK}) = \frac{\lambda_t^R\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)}{2(1-\rho_R^{ACK})}. \quad (4.109)$$

**Gesamtverzögerung:** Nachdem die mittleren Wartezeiten einer Nachricht bis zur Bearbeitung beim Sender und Empfänger bestimmt wurden, kann die Verzögerung einer bestimmten Nachricht ermittelt werden. Die Gesamtverzögerung beinhaltet die Wartezeit einer Datennachricht nach der Generierung bzw. dem Empfang beim Sender, die Zeit um die Nachricht zu senden, die Verzögerung im Netzwerk, die Zeit um die Nachricht beim Empfänger zu empfangen und die Wartezeit bis zur Bearbeitung.

$T_S$ sei das Zeitgeberintervall beim Sender, nach dessen Ablauf eine nichtquittierte Nachricht als verloren angenommen und eine Übertragungswiederholung ausgelöst wird. $\tau$ sei die Verzögerung im Netzwerk, um eine Nachricht zwischen zwei beliebigen Rechnern auszutauschen. Die Wahrscheinlichkeit, dass eine Datennachricht $j$ Übertragungen bis zum korrekten Empfang benötigt, ist $P(M_r=j)=q_D^{j-1}(1-q_D)$. $M_r$ ist die notwendige Anzahl von Übertragungen bis zum korrekten Empfang einer Datennachricht bei einem beliebigen Empfänger $r$. Jetzt kann der Erwartungswert der Durchschnittsverzögerung $E(S_\phi^{ACK})$ für eine Datennachricht bestimmt werden (siehe auch [Yamamoto et al. 1997]). Allgemein ist der Erwartungswert die Summe über alle möglichen Ereignisse multipliziert mit der Wahrscheinlichkeit ihres Eintritts. In diesem Fall ergibt sich eine unendliche Summe über der Anzahl an Übertragungen $j$:

$$E(S_\phi^{ACK}) = \left[\sum_{j=1}^{\infty} q_D^{j-1}(1-q_D)\Big(j\Big(E(W_S^{ACK})+E(X)\Big)+(j-1)T_S\Big)\right]+\tau+E(W_R^{ACK})+E(X). \quad (4.110)$$

Für jede Übertragung $j$ muss zuerst eine Wartezeit beim Sender berücksichtigt werden, $E(W_S^{ACK})$, bis die Nachricht gesendet werden kann. Die Sendezeit beträgt $E(X)$. Nach jeder erfolglosen Übertragung, d.h. einer Übertragung die nicht zum Empfang aller positiven Quittungen beim Sender führt, muss das Zeitgeberintervall $T_S$ zum Sammeln der Quittungen abgewartet werden, bevor eine weitere Übertragungswiederholung gesendet werden kann. Einzig für die letzte, erfolgreiche Übertragungswiederholung ist der Zeitgeber bedeutungslos. Für diese letzte Übertragungswiederholung wird die Zeit im Netzwerk, $\tau$, berücksichtigt, die Wartezeit des Empfängers, bis sie empfangen werden kann, $E(W_R^{ACK})$ und die Zeit zum Empfangen der Nachricht, $E(X)$. Mit Hilfe des Erwartungswerts der geometrischen Verteilung (siehe Formel 4.2 und Formel 4.3) lässt sich die unendliche Summe beseitigen:

$$E(S_\phi^{ACK}) = \frac{E(W_S^{ACK})+E(X)+q_D T_S}{1-q_D}+\tau+E(W_R^{ACK})+E(X). \quad (4.111)$$

$\frac{1}{1-q_D}$ entspricht der notwendigen Anzahl von Übertragungen für einen einzelnen Empfänger, $E(M_r)$ (siehe Formel 4.4). Durch die Abspaltung des ersten Summanden aus der Berechnung des Erwartungswerts ergibt sich, dass $E(M_r^{ACK})-1=\frac{q_D}{1-q_D}$ ist. Damit kann die Berechnung der Auslieferungsverzögerung weiter vereinfacht werden zu:

$$\begin{aligned} E(S_\phi^{ACK}) &= \underbrace{E(M_r^{ACK})\Big(E(W_S^{ACK})+E(X)\Big)+\Big(E(M_r^{ACK})-1\Big)T_S}_{\textit{Zeit beim Sender}} \\ &+ \underbrace{\tau}_{\textit{Zeit im Netzwerk}} + \underbrace{E(W_R^{ACK}+E(X)).}_{\textit{Zeit beim Empfänger}} \end{aligned} \tag{4.112}$$

Nachdem die Durchschnittsverzögerung ermittelt wurde, soll die Maximalverzögerung ermittelt werden. Zur Erinnerung sei gesagt, dass bei der Maximalverzögerung $E(S_\gamma^{ACK})$ die Wahrscheinlichkeit angegeben wird, mit der alle Empfänger die Datennachricht erfolgreich erhalten. Diese Wahrscheinlichkeit wird mit $\gamma$ bezeichnet. Wenn $E(M_\gamma^{ACK})$ die notwendige Anzahl an Übertragungen bezeichnet, dann gilt: $\gamma=1-{q_D}^{E(M_\gamma^{ACK})}$. Die notwendige Anzahl an Übertragungen kann aus dem Parameter $\gamma$ und schließlich $E(S_\gamma^{ACK})$ bestimmt werden:

$$E(M_\gamma^{ACK}) = \frac{ln(1-\gamma)}{ln(q_D)}, \qquad \gamma \geq 1-q_D \tag{4.113}$$

$$E(S_\gamma^{ACK}) = E(M_\gamma^{ACK})\Big(E(W_S^{ACK})+E(X_\gamma^{ACK})\Big)+\Big(E(M_\gamma^{ACK})-1\Big)T_S+\tau+E(W_R^{ACK})+E(X). \tag{4.114}$$

Die mittlere Umlaufverzögerung, $E(S_{RTD}^{ACK})$, beschränkt für eine fensterbasierte Flusskontrolle den maximalen Durchsatz. Anstatt wie in Formel 4.112 nur einen beliebigen Empfänger zu betrachten, ist für deren Berechnung die notwendige Anzahl an Übertragungen für alle Empfänger, $M^{ACK}$, zu verwenden:

$$\begin{aligned} E(S_{RTD}^{ACK}) &= \underbrace{\Big(E(M^{ACK})-1\Big)\Big(T_S+E(W_S^{ACK})+E(X)\Big)}_{\textit{Übertragungen senden}} \\ &\underbrace{+E(X)+E(W_S^{ACK})+\tau+E(W_R^{ACK})+E(X)}_{\textit{letzte Übertragung senden und empfangen}} \\ &\underbrace{+E(Y)+E(W_R^{ACK})+\tau+E(W_S^{ACK})+E(Y).}_{\textit{letzte Quittung senden und empfangen}} \end{aligned} \tag{4.115}$$

#### 4.5.4.2 Empfängerinitiiertes Protokoll (K-NACK)

Alle Verzögerungskomponenten dieser Protokollklasse sind beispielhaft in Abbildung 4.9 dargestellt.

**Mittlere Wartezeit einer Nachricht beim Sender:** Der Sender hat die folgenden Nachrichtenflüsse zu bearbeiten:

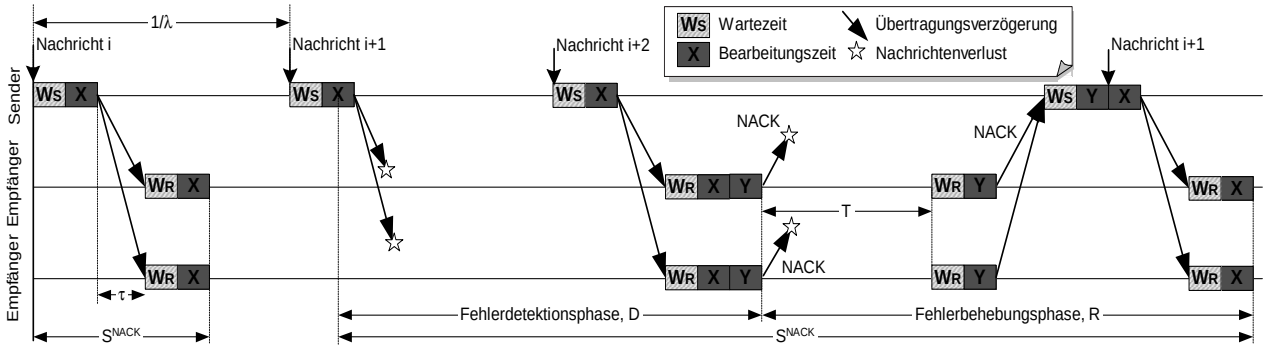



*Abbildung 4.9: Verzögerungskomponenten der Protokollklasse K-NACK*

1. Fluss $\lambda_t^S$ der Datennachrichten, die zum ersten Mal übertragen werden. Die Rate beträgt $\lambda$ und die Bearbeitungszeit für eine Datennachricht ist $X$.

2. Eingehender Nachrichtenfluss der negativen Quittungen $\lambda_r^S$ mit Rate $\lambda(E(M^{NACK})-1)$, die eine Übertragungswiederholung auslösen. Folglich ist die Bearbeitungszeit $X+Y$.

3. Der letzte Nachrichtenfluss $\lambda_n^S$ mit Rate $\lambda\Big(E(L^{NACK})(1-p_N)-(E(M^{NACK})-1)\Big)$ ergibt sich aus den zusätzlichen negativen Quittungen, die empfangen und verarbeitet werden müssen, aber keine Übertragungswiederholung auslösen. Die Bearbeitungszeit ist $Y$. $E(L^{NACK})$ ist die Gesamtanzahl gesendeter negativer Quittungen. Die Berechnung ist in Formel 4.30 gegeben.

Mit diesen Nachrichtenflüssen ergibt sich aufgrund von Formel 4.103 die folgende Last des Senders:

$$\begin{aligned}\rho_S^{NACK} &= \lambda E(X)+\lambda\Big(E(M^{NACK})-1\Big)E(X+Y)+\lambda\Big[E(L^{NACK})(1-p_N)-\Big(E(M^{NACK})-1\Big)\Big]E(Y)\\ &= \lambda E(M^{NACK})E(X)+\lambda E(L^{NACK})(1-p_N)E(Y). \qquad (4.116)\end{aligned}$$

Mit der Last und unter Zuhilfenahme von Formel 4.104 lässt sich die mittlere Wartezeit für eine Nachricht bis zur Bearbeitung bestimmen:

$$E(W_S^{NACK}) = \frac{\lambda_t^S E(X^2)+\lambda_r^S\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)+\lambda_n^S E(Y^2)}{2(1-\rho_S^{NACK})}. \qquad (4.117)$$

**Mittlere Wartezeit einer Nachricht beim Empfänger:** Beim Empfänger ergeben sich die folgenden zwei Nachrichtenflüsse:

1. Der Fluss empfangener Übertragungen vom Sender $\lambda_t^R$ mit der Rate $\lambda E(M^{NACK})(1-q_D)$ und Bearbeitungszeit $X$.

2. Auf Datenverlust wird mit negativen Quittungen geantwortet. $\lambda_n^R$ ist der Fluss negativer Quittungen mit Rate $\lambda\left(E(O_r^{NACK})-1\right)$ und Bearbeitungszeit $Y$. $O_r^{NACK}$ ist die Anzahl notwendiger Übertragungen für einen einzelnen Empfänger inklusive der leeren Runden, in denen aufgrund von Quittungsverlust keine Übertragungswiederholungen gesendet werden. Die Berechnung von $O_r^{NACK}$ ist in Formel 4.25 gegeben.

Die Last des Empfängers ist:

$$\rho_R^{NACK} = \lambda E(M^{NACK})(1-q_D)E(X)+\lambda\left(E(O_r^{NACK})-1\right)E(Y). \quad (4.118)$$

Die mittlere Wartezeit einer Nachricht bis zur Bearbeitung ist:

$$E(W_R^{NACK}) = \frac{\lambda_t^R E(X^2)+\lambda_n^R E(Y^2)}{2(1-\rho_R^{NACK})}. \quad (4.119)$$

**Gesamtverzögerung:** Zu beachten ist, dass ein Empfänger Datenverlust durch eine Lücke in den Sequenznummern der Datennachrichten feststellt, d.h. wenn Nachricht $i+1$ empfangen wurde ohne zuvor Nachricht $i$ empfangen zu haben. Die Konsequenz aus diesem Fehlerentdeckungsmechanismus ist, dass die Zeit bis zur Fehlerentdeckung von der Senderate $\lambda$ abhängt.

Nachdem ein Empfänger Datenverlust entdeckt und eine negative Quittung gesendet hat, wird ein Zeitgeber gestartet. Nach dessen Ablauf wird die negative Quittung wiederholt und der Zeitgeber erneut gestartet, bis die Datennachricht erfolgreich empfangen wurde. Auch der Sender startet nach dem Senden jeder Übertragung einen Zeitgeber. Während dieses Intervalls sammelt der Sender alle eingehenden negativen Quittungen und sendet nach Ablauf des Zeitgebers eine gemeinsame Übertragungswiederholung.

Zur Bestimmung der Gesamtverzögerung wird in [Yamamoto et al. 1997] vorgeschlagen, zwischen folgenden beiden Phasen zu unterscheiden:

1. Fehlerdetektionsphase. Diese Phase beinhaltet die Zeitspanne zwischen dem initialen Eintreffen einer Nachricht beim Sender und dem Auslösen der ersten negativen Quittung bei einem beliebigen Empfänger, der die Datennachricht $i$ verloren hat. Der Verlust wird mit dem korrekten Empfang einer Datennachricht $j$ detektiert, wobei $j > i$ ist.

2. Fehlerbehebungsphase. Diese Phase umfasst die Zeitspanne zwischen dem Ende der Fehlerdetektionsphase und dem korrekten Empfang der Datennachricht auf dem betreffenden Empfänger. Da Datennachrichten und auch Quittungen verloren gehen können, umfasst die Fehlerbehebungsphase das periodische Senden von negativen Quittungen, bis die Datennachricht erfolgreich empfangen wird.

Die Berechnung der Phasen ist im Folgenden gegeben und unterscheidet sich aufgrund des hier verwendeten erweiterten Systemmodells von [Yamamoto et al. 1997].

**Fehlerdetektionsphase:** Für die Bestimmung der Fehlerdetektionsphasendauer muss die Zeit für erfolglose Übertragungen aufeinanderfolgender Datennachrichten, die Zeit für die erste erfolgreiche Datennachricht und die Zeit für das Senden der ersten negativen Quittung für die erste verlorene Datennachricht berücksichtigt werden. Die Zufallsvariable $L$ bezeichnet die Anzahl aufeinanderfolgend verlorener Nachrichten bei $k+1$ erfolglosen Empfängern. Die Wahrscheinlichkeitsverteilung von $L$ ist wie folgt gegeben:

$$P(L=l|K=k) \quad = \quad {q_D}^{l(k+1)}(1-{q_D}^{k+1}), \quad l=0,1,\ldots \text{ und } k=0,1,\ldots,R-1. \tag{4.120}$$

${q_D}^{l(k+1)}$ ist die Wahrscheinlichkeit bei $k+1$ Empfängern eine Nachricht $l$-mal nicht zu empfangen. Dementsprechend ist $1-{q_D}^{k+1}$ die Wahrscheinlichkeit für den erfolgreichen Nachrichtenempfang bei mindestens einem Empfänger. Der Erwartungswert der Anzahl aufeinanderfolgend verlorener Nachrichten bei $k+1$ Empfängern folgt aus dem Erwartungswert der geometrischen Verteilung (siehe Formel 4.2 und Formel 4.3):

$$E(L|K=k) \quad = \quad \frac{{q_D}^{k+1}}{1-{q_D}^{k+1}}. \tag{4.121}$$

Damit keine bestimmte Anzahl von Empfängern festgelegt werden muss, die alle gemeinsam nicht erfolgreich sind, soll das mittlere $k$ aus den möglichen zwischen 0 und $R-1$ bestimmt werden:

$$E(L|K) \quad = \quad \sum_{k=0}^{R-1}\binom{R-1}{k}{q_D}^{k}(1-q_D)^{R-1-k}\frac{{q_D}^{k+1}}{1-{q_D}^{k+1}}. \tag{4.122}$$

Nun wird die mittlere Anzahl aufeinanderfolgend verlorener Nachrichten mit der Zeit $\frac{1}{\lambda}$ multipliziert die vergeht, um eine Nachricht entsprechend der Senderate zu empfangen. Die Verzögerung der Nachricht mit der Nummer $L+1$, d.h. der ersten erfolgreichen nach $L$ erfolglosen Nachrichten, ist somit:

$$\underbrace{E(W_S^{NACK})+E(X)}_{\textit{Verzögerung beim Sender}} \quad + \quad \underbrace{\tau}_{\textit{Verzögerung im Netzwerk}} \quad + \quad \underbrace{E(W_R^{NACK})+E(X).}_{\textit{Verzögerung beim Empfänger}} \tag{4.123}$$

Die gesamte Fehlerdetektionsphase kann wie folgt berechnet werden werden:

$$\begin{aligned} E(D^{NACK}) \quad = \quad & \sum_{k=0}^{R-1}\binom{R-1}{k}{q_D}^{k}(1-q_D)^{R-1-k}\frac{{q_D}^{k+1}}{1-{q_D}^{k+1}}\frac{1}{\lambda} \\ & + E(W_S^{NACK})+E(W_R^{NACK})+\tau+2E(X)+E(Y). \end{aligned} \tag{4.124}$$

**Fehlerbehebungsphase:** Aus der Sichtweise eines beliebigen Empfängers besteht diese Phase aus einer bestimmten Anzahl von Runden, in denen erfolglos eine Übertragungswiederholung angefordert wird. Nach der initialen negativen Quittung der Fehlerdetektionsphase haben alle folgenden Runden die Verzögerung $T_R + E(W_R^{NACK}) + E(Y)$, wobei $T_R$ das Zeitgeberintervall bestimmt, $W_R^{NACK}$ die Wartezeit beim Empfänger und $Y$ die Bearbeitungszeit, um eine negative Quittung zu senden. Nach einer Reihe erfolgloser Quittungen endet die Fehlerbehebungsphase letztlich mit einer erfolgreichen negativen Quittung und einer anschließenden erfolgreichen Übertragungswiederholung. Dies beinhaltet die Verzögerung im Netzwerk für die negative Quittung, die Wartezeit und die Bearbeitungszeit beim Sender, die Verzögerung der Datennachricht im Netzwerk auf dem Weg zum Empfänger und die Bearbeitungs– und Wartezeit beim Empfänger. Die mittlere Zeit für die Fehlerbehebungsphase, vorerst unter der Annahme, dass negative Quittungen nicht verloren gehen, ist gegeben durch:

$$\begin{aligned} E(R^{NACK}) &= \sum_{j=2}^{\infty} {q_D}^{j-1}(1-q_D)(j-2)\Big(T_R + E(W_R^{NACK}) + E(Y)\Big) \\ &\quad + 2\Big(E(X)+\tau\Big) + E(W_S^{NACK}) + E(W_R^{NACK}) + E(Y) \qquad (4.125) \\ &= \frac{{q_D}^2}{1-q_D}\Big(T_R + E(W_R^{NACK}) + E(Y)\Big) + 2\Big(E(X)+\tau\Big) \\ &\quad + E(W_S^{NACK}) + E(W_R^{NACK}) + E(Y) \qquad (4.126) \\ &= \Big(E(M_r^{NACK}) - 2\Big)\Big(T_R + E(W_R^{NACK}) + E(Y)\Big) + 2\Big(E(X)+\tau\Big) \\ &\quad + E(W_S^{NACK}) + E(W_R^{NACK}) + E(Y). \qquad (4.127) \end{aligned}$$

$j$ ist die Anzahl erfolgloser Runden aufgrund des Datenverlusts. Die Berücksichtigung von Quittungsverlust ergibt einen wichtigen Sonderfall, den Verlust aller gesendeten Quittungen. Nur dieser Sonderfall erhöht die benötigte Zeit für die Fehlerbehebungsphase. Im Gegensatz zu $M_r^{NACK}$ berücksichtigt $O_r^{NACK}$ (siehe Formel 4.25) die Gesamtanzahl notwendiger Übertragungsrunden inklusive der leeren Runden, in denen alle Quittungen verloren gehen. Damit ändert sich die obige Gleichung unter Berücksichtigung von Quittungsverlust zu:

$$\begin{aligned} E(R^{NACK}) &= \Big(E(O_r^{NACK}) - 2\Big)\Big(T_R + E(W_R^{NACK}) + E(Y)\Big) + 2\Big(E(X)+\tau\Big) \\ &\quad + E(W_S^{NACK}) + E(W_R^{NACK}) + E(Y). \qquad (4.128) \end{aligned}$$

Die Durchschnittsverzögerung, $E(S_\phi^{NACK})$, der Protokollklasse K-NACK besteht im Fehlerfall ($q_D$) aus beiden Phasen, der Fehlerdetektionsphase und der Fehlerbehebungsphase. Für den Fall, dass kein Fehler auftritt ($1-q_D$), müssen keine Übertragungswiederholungen berücksichtigt werden:

$$E(S_\phi^{NACK}) = (1-q_D)\Big(E(W_S^{NACK}) + E(W_R^{NACK}) + 2E(X) + \tau\Big) + q_D\Big(E(D^{NACK}) + E(R^{NACK})\Big). \qquad (4.129)$$

Die Maximalverzögerung, $E(S_\gamma^{NACK})$, um alle Datennachrichten mit einer bestimmten Wahrscheinlichkeit $\gamma$ erfolgreich zu empfangen, kann mit einer leicht modifizierten Fehlerbehebungsphase ermittelt werden:

$$\begin{aligned} E(R^{NACK}) &= \Big(E(M_\gamma^{NACK}) + E(O_{e,r}^{NACK}) - 2\Big)\Big(T_R + E(W_R^{NACK}) + E(Y)\Big) \\ &\quad + \tau + E(W_S^{NACK}) + E(Y) + E(X) + \tau + E(W_R^{NACK}) + E(X) \qquad (4.130) \\ E(S_\gamma^{NACK}) &= E(D^{NACK}) + E(R^{NACK}). \qquad (4.131) \end{aligned}$$

$E(O_{e,r}^{NACK})$ wird analog zu Formel 4.29 mit $E(M_\gamma^{NACK})$ anstatt $E(M_r^{NACK})$ als obere Schranke in der Summe bestimmt.

Wie bereits in der Protokollklasse K-ACK soll abschließend die Umlaufverzögerung bestimmt werden. Allerdings bietet diese Protokollklasse analog zu K-NAPP dem Sender keine Signalisierung für den erfolgreichen Empfang einer Datennachricht bei allen Empfängern. Deshalb soll für die Berechnung der Umlaufverzögerung die Annahme getroffen werden, dass eine derartige Signalisierung mit positiven Quittungen dennoch gegeben sei. Reale Protokolle müssen diese verwenden, wenn sie eine fensterbasierte Flusskontrolle einsetzen. Die Signalisierung erfolgt dann häufig periodisch, nachdem eine festgelegte Anzahl von Nachrichten empfangen wurde. Gegenüber einer Bestätigung nach jedem Empfang wird ein geringerer Aufwand erzeugt. Auch ohne eine explizite Signalisierung ist die Bestimmung der Umlaufverzögerung sinnvoll. Der Sender muss einen Zeitgeber verwalten, nach dessen Ablauf er annimmt, dass alle Empfänger die Daten erfolgreich empfangen haben und diese gelöscht werden können. Die Bestimmung der Umlaufverzögerung gibt einen Anhaltspunkt für die sinnvolle Zeitgeberbelegung, da nach dieser Zeit keine negativen Quittungen mehr erwartet werden. Die Umlaufverzögerung beträgt für die Protokollklasse K-NACK:

$$\begin{aligned} E(R^{NACK}) &= \Big(E(O^{NACK}) - 2\Big)\Big(T_R + E(W_R^{NACK}) + E(Y)\Big) + 2\Big(E(X) + \tau\Big) \\ &\quad + E(W_S^{NACK}) + E(W_R^{NACK}) + E(Y) \qquad (4.132) \\ E(S_{RTD}^{NACK}) &= (1-q_D)^R\Big(E(W_S^{NACK}) + E(W_R^{NACK}) + 2E(X) + \tau\Big) \\ &\quad + \Big(1-(1-q_D)^R\Big)\Big(E(D^{NACK}) + E(R^{NACK})\Big) + E(Y) + \tau + E(W_S^{NACK}) + E(Y). \qquad (4.133) \end{aligned}$$

$(1-q_D)^R$ ist die Wahrscheinlichkeit dafür, dass alle Empfänger die Daten korrekt empfangen haben und demnach keine Fehlerdetektions– und Fehlerbehebungsphase notwendig ist.

#### 4.5.4.3 Empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-NAPP)

**Mittlere Wartezeit einer Nachricht beim Sender:** Die Verzögerungskomponenten sind in Abbildung 4.10 dargestellt. Beim Sender lassen sich die folgenden drei Nachrichtenflüsse unterscheiden:

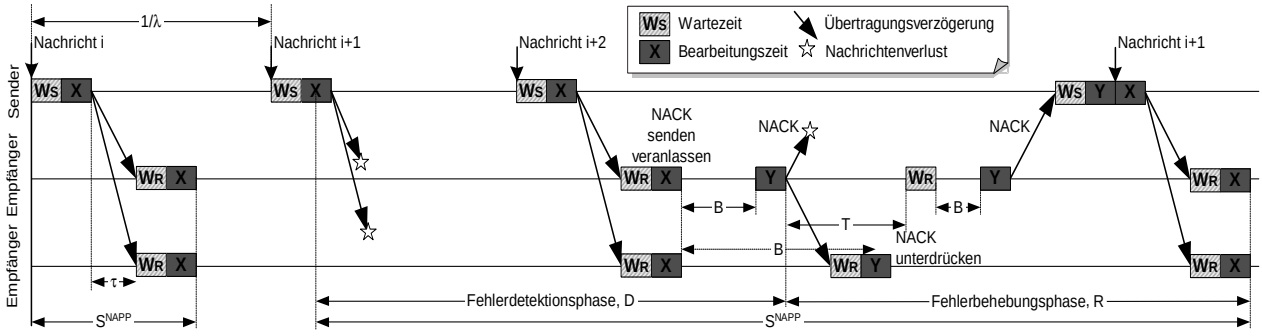



*Abbildung 4.10: Verzögerungskomponenten der Protokollklasse K-NAPP*

1. Der Nachrichtenfluss der initialen Übertragung $\lambda_t^S$ mit Rate $\lambda$ und Bearbeitungszeit $X$.

2. Der Nachrichtenfluss negativer Quittungen, die eine Übertragungswiederholung auslösen. Dieser Fluss wird mit $\lambda_r^S$ bezeichnet, hat die Rate $\lambda\Big(E(M^{NAPP})-1\Big)$ und die Bearbeitungszeit $X+Y$, da auf eine empfangene negative Quittung eine Übertragungswiederholung folgt.

3. Schließlich der Fluss zusätzlicher negativer Quittungen $\lambda_n^S$ mit Rate $\lambda E(L^{NAPP})(1-q_N)-\lambda_r^S$ und Bearbeitungszeit $Y$, die vom Sender empfangen und verarbeitet werden müssen, aber keine Übertragungswiederholung auslösen. $L^{NAPP}$ bezeichnet die Gesamtanzahl gesendeter negativer Quittungen (siehe Formel 4.42).

Mit diesen Nachrichtenflüssen, die der Sender zu bearbeiten hat, ergibt sich folgende Last:

$$\rho_S^{NAPP} = \lambda E(M^{NAPP})E(X)+\lambda E(L^{NAPP})(1-q_N)E(Y). \tag{4.134}$$

Mit Hilfe von Formel 4.104 ergibt sich die mittlere Wartezeit einer Nachricht bis zur Bearbeitung beim Sender zu:

$$E(W_S^{NAPP}) = \frac{\lambda_t^S E(X^2)+\lambda_r^S\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)+\lambda_n^S E(Y^2)}{2(1-\rho_S^{NAPP})}. \tag{4.135}$$

**Mittlere Wartezeit einer Nachricht beim Empfänger:** Die Nachrichtenflüsse des Empfängers lassen sich unterscheiden in die folgenden drei:

1. Der Nachrichtenfluss der Datenübertragungen vom Sender $\lambda_t^R$ mit Rate $\lambda E(M^{NAPP})(1-q_D)$ und Bearbeitungszeit $X$.

2. Die gesendeten negativen Quittungen werden mit dem Fluss $\lambda_n^R$ bezeichnet. Die Rate der negativen Quittungen ist $\lambda\Big(E(O_r^{NAPP})-1\Big)E(L_\phi^{NAPP})/E(R_N)$ und Bearbeitungszeit $Y$. Die komplizierte Rate der negativen Quittungen wurde bereits in Abschnitt 4.5.2.3 berechnet.

3. Schließlich werden zudem negative Quittungen anderer Empfänger empfangen. Dieser Nachrichtenfluss wird mit $\lambda_{n,r}^R$ bezeichnet und hat die Bearbeitungszeit $Y$. Die Erläuterung zur Berechnung der Rate ist ebenfalls in Abschnitt 4.5.2.3 gegeben. Sie setzt sich zusammen aus allen gesendeten Quittungen abzüglich der eigenen und ist folglich $\lambda(1-q_N)\Big[E(L_\phi^{NAPP})\Big(E(O^{NAPP})-1\Big)-\Big(E(L_\phi^{NAPP})/E(R_N)\Big)\Big(E(O_r^{NAPP})-1\Big)\Big]$.

Die Summe der Nachrichtenflüsse ergibt die Last des Empfängers:

$$\rho_R^{NAPP} = \lambda_t^R E(X)+(\lambda_n^R+\lambda_{n,r}^R)E(Y). \tag{4.136}$$

Die mittlere Wartezeit einer Nachricht bis zur Bearbeitung beim Empfänger ist:

$$E(W_R^{NAPP}) = \frac{\lambda_t^R E(X^2)+(\lambda_n^R+\lambda_{n,r}^R)E(Y^2)}{2(1-\rho_R^{NAPP})}. \tag{4.137}$$

Wie bereits in der vorigen Protokollklasse geschehen, ist es bei empfängerinitiierten Protokollen sinnvoll, zwischen der Fehlerdetektionsphase und der Fehlerbehebungsphase zu unterscheiden.

**Fehlerdetektionsphase:** Diese Phase umfasst die Zeitspanne zwischen dem initialen Eintreffen oder der Generierung einer Nachricht beim Sender und der ersten gesendeten negativen Quittung. Im Gegensatz zur Protokollklasse K-NACK muss zusätzlich eine zufällige Verzögerungszeit $E(B^{NAPP})$ berücksichtigt werden. Zur Erinnerung: Die Quittungsunterdrückung beruht auf dem Prinzip, dass das Senden der negativen Quittung um eine zufällige Zeit verzögert wird, während die Empfänger auf eingehende Quittungen warten, um das mehrfache Senden einer Quittung zu verhindern. Die Verzögerungszeit beginnt mit der Entdeckung von Nachrichtenverlust und endet mit dem Ablauf eines Zeitgebers und dem Senden der ersten negativen Quittung. Die Berechnung der Fehlerdetektionsphase ist bis auf $E(B^{NAPP})$ identisch zur Protokollklasse K-NACK:

$$\begin{aligned} E(D^{NAPP}) = {} & \underbrace{\sum_{k=0}^{R-1}\binom{R-1}{k}{q_D}^k(1-q_D)^{R-1-k}\frac{{q_D}^{k+1}}{1-{q_D}^{k+1}}}_{\textit{aufeinanderfolgend verlorene Nachrichten}}\ \frac{1}{\lambda} \\ & + \underbrace{E(W_S^{NAPP})+E(W_R^{NAPP})+\tau+2E(X)}_{\textit{Verzögerung des ersten Empfangs}} + E(B^{NAPP})+E(Y). \end{aligned} \tag{4.138}$$

**Fehlerbehebungsphase:** Die Fehlerbehebungsphase für K-NAPP ist ebenfalls bis auf $E(B^{NAPP})$ identisch zur Protokollklasse K-NACK:

$$\begin{aligned} E(R^{NAPP}) = {} & \Big(E(O_r^{NAPP})-2\Big)\Big(T_R+E(W_R^{NAPP})+E(B^{NAPP})+E(Y)\Big) \\ & +2\Big(E(X)+\tau\Big)+E(Y)+E(W_S^{NAPP})+E(W_R^{NAPP}). \end{aligned} \tag{4.139}$$

**Gesamtverzögerung:** Die durchschnittliche Gesamtverzögerung der Protokollklasse K-NAPP ergibt sich aus der Fehlerdetektions– und Fehlerbehebungsphase im Fall von Datenverlust oder der direkten Übertragung mit Wahrscheinlichkeit $1-q_D$ für keinen Datenverlust:

$$E(S_\phi^{NAPP}) = (1-q_D)\Big(E(W_S^{NAPP})+E(W_R^{NAPP})+2E(X)+\tau\Big)+q_D\Big(E(D^{NAPP})+E(R^{NAPP})\Big). \quad (4.140)$$

Die Maximalverzögerung ist:

$$\begin{aligned} E(R^{NAPP}) &= \Big(E(M_\gamma^{NAPP})+E(O_{e,r}^{NAPP})-2\Big)\Big(T_R+E(W_R^{NAPP})+E(B^{NAPP})+E(Y)\Big) \\ &\quad +\tau+E(W_S^{NAPP})+E(Y)+E(X)+\tau+E(W_R^{NAPP})+E(X) \quad (4.141) \\ E(S_\gamma^{NAPP}) &= E(D^{NAPP})+E(R^{NAPP}). \quad (4.142) \end{aligned}$$

Die Anmerkungen zur vorigen Protokollklasse für die Umlaufverzögerung gelten hier analog, d.h. es ist keine Signalisierung vorhanden. Dennoch ist es vorteilhaft die Umlaufverzögerung zu bestimmen, z.B. für die korrekte Einstellung der Zeitgeber. Die Umlaufverzögerung ist:

$$\begin{aligned} E(R^{NAPP}) &= \Big(E(O^{NAPP})-2\Big)\Big(T_R+E(W_R^{NAPP})+E(B^{NAPP})+E(Y)\Big) \\ &\quad +2\Big(E(X)+\tau\Big)+E(Y)+E(W_S^{NAPP})+E(W_R^{NAPP}) \quad (4.143) \\ E(S_{RTD}^{NAPP}) &= (1-q_D)^R\Big(E(W_S^{NAPP})+E(W_R^{NAPP})+2E(X)+\tau\Big) \\ &\quad +\Big(1-(1-q_D)^R\Big)\Big(E(D^{NAPP})+E(R^{NAPP})\Big)+E(Y)+\tau+E(W_S^{NAPP})+E(Y). \quad (4.144) \end{aligned}$$

#### 4.5.4.4 Hierarchisches senderinitiiertes Protokoll (K-HACK)

**Mittlere Wartezeit einer Nachricht beim Sender:** Abbildung 4.11 stellt die Verzögerungskomponenten dar, die in der Analyse Berücksichtigung finden. Der Sender hat die folgenden Nachrichtenflüsse zu bearbeiten:

1. Den Nachrichtenfluss der initialen Datennachrichten $\lambda_t^S$ mit Rate $\lambda$ und Bearbeitungszeit $X$.
2. Den Nachrichtenfluss der Übertragungswiederholungen $\lambda_r^S$ mit Rate $\lambda(E(M^{HACK})-1)$, da jede Nachricht $(E(M^{HACK})-1)$-mal wiederholt wird.
3. Den Nachrichtenfluss empfangener Quittungen $\lambda_a^S$. Dieser besitzt die Rate $\lambda BE(M^{HACK})(1-q_D)(1-p_A)$. Zur Erinnerung: $B$ ist der Verzweigungsgrad des Kontrollbaums. Die Bearbeitungszeit für eine Quittung ist $Y$.

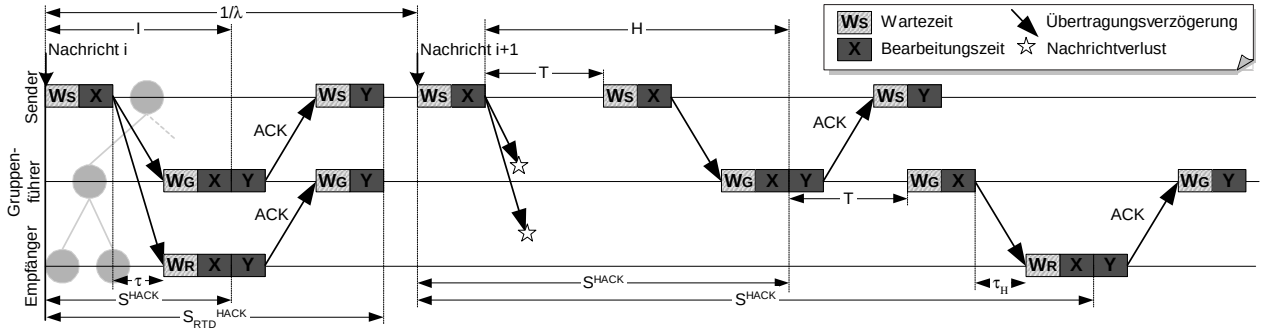



*Abbildung 4.11: Verzögerungskomponenten der Protokollklasse K-HACK*

Die Last des Senders, $\rho_S^{HACK}$, ist die Summe der Nachrichtenflüsse mit ihren Bearbeitungszeiten:

$$\rho_S^{HACK} = \lambda E(M^{HACK})E(X) + \lambda BE(M^{HACK})(1-q_D)(1-p_A)E(Y). \tag{4.145}$$

Aus Formel 4.104 ergibt sich die mittlere Wartezeit:

$$E(W_S^{HACK}) = \frac{(\lambda_t^S + \lambda_r^S)E(X^2) + \lambda_a^S E(Y^2)}{2(1-\rho_S^{HACK})}. \tag{4.146}$$

**Mittlere Wartezeit einer Nachricht beim Empfänger:** Beim Empfänger gibt es zwei Nachrichtenflüsse bestehend aus ankommenden Übertragungen. Jede ankommende Übertragung des Vaterknotens im Kontrollbaum wird mit einer Quittung bestätigt. Der Nachrichtenfluss wird mit $\lambda_t^R$ bezeichnet und hat die Rate $\lambda E(M^{HACK})(1-q_D) = \lambda E(\widetilde{N}_r^{HACK})$. Die Bearbeitungszeit ist $X+Y$. Zusätzlich empfangene Übertragungswiederholungen anderer lokaler Gruppen verursachen den Nachrichtenfluss $\lambda_G^R$ mit der Rate $\lambda\Big(E(\widetilde{N}_{r,t}^{HACK}) - E(\widetilde{N}_r^{HACK})\Big)$. Da diese nicht mit einer Quittung bestätigt werden, ist deren Bearbeitungszeit lediglich $X$. Die Last des Empfängers ist:

$$\rho_R^{HACK} = \lambda E(\widetilde{N}_r^{HACK})\Big(E(X)+E(Y)\Big) + \lambda\Big(E(\widetilde{N}_{r,t}^{HACK}) - E(\widetilde{N}_r^{HACK})\Big)E(X). \tag{4.147}$$

Die mittlere Wartezeit einer Nachricht beim Empfänger bis zur Bearbeitung beträgt:

$$E(W_R^{HACK}) = \frac{\lambda_t^R E\Big((X+Y)^2\Big) + \lambda_G^R E(X^2)}{2(1-\rho_R^{HACK})}. \tag{4.148}$$

Unter der Annahme $X$ und $Y$ seien unabhängige Zufallsvariablen gilt $E(X_1+X_2) = E(X_1)+E(X_2)$, $Var(X_1) = E(X_1^2) - (E(X_1))^2$ und $Var(X_1+X_2) = Var(X_1)+Var(X_2)$. Daraus folgt $E(W_R^{HACK})$ zu:

$$E(W_R^{HACK}) = \frac{\lambda_t^R\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big) + \lambda_G^R E(X^2)}{2(1-\rho_R^{HACK})}. \tag{4.149}$$

**Mittlere Wartezeit einer Nachricht beim Gruppenführer:** Die Last eines Gruppenführers (außer dem Sender) besteht aus der Last des Senders abzüglich der initialen Übertragung und der Last eines Empfängers:

$$\begin{aligned}\rho_G^{HACK} &= \underbrace{\lambda\Big(E(M^{HACK})-1\Big)}_{\lambda_r^S}E(X)+\underbrace{\lambda BE(M^{HACK})(1-q_D)(1-p_A)}_{\lambda_a^S}E(Y)\\ &\quad+\underbrace{\lambda E(M^{HACK})(1-q_D)}_{\lambda_t^R}\Big(E(X)+E(Y)\Big)+\underbrace{\lambda\Big(E(\widetilde{N}_g^{HACK})-E(\widetilde{N}_r^{HACK})\Big)}_{\lambda_G^R}E(X). \quad (4.150)\end{aligned}$$

Die mittlere Wartezeit beträgt:

$$E(W_G^{HACK}) = \frac{\lambda_r^S E(X^2)+\lambda_a^S E(Y^2)+\lambda_t^R\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)+\lambda_G^R E(X^2)}{2(1-\rho_G^{HACK})}. \quad (4.151)$$

**Gesamtverzögerung:** $T$ ist das Zeitgeberintervall, innerhalb dessen Gruppenführer die Quittungen ihrer Kindknoten sammeln. Werden nicht alle Quittungen empfangen, dann wird nach Ablauf des Zeitgebers eine Übertragungswiederholung gesendet. $\tau$ ist die Nachrichtenverzögerung im Netzwerk, $\overline{h}$ die durchschnittliche Anzahl von Hierarchiestufen im Kontrollbaum (siehe Formel 4.51) und $B$ der Verzweigungsgrad. Wenn keine Übertragungswiederholung notwendig ist, dann beträgt die Verzögerung der initialen Übertragung, $E(I)$:

$$E(I) = E(W_S^{HACK})+E(X)+\tau+E(W_G^{HACK})+E(X). \quad (4.152)$$

Hier wird eine vereinfachende pessimistische Annahme getroffen, indem die längere mittlere Wartezeit eines Gruppenführers anstatt eines Empfängers verwendet wird. Nun wird die Verzögerung für eine Übertragungswiederholung im Kontrollbaum ermittelt. Die Zeit für eine hierarchische Übertragungswiederholung, $E(H)$, unter der Annahme, dass der Vaterknoten die zu wiederholende Nachricht korrekt empfangen hat, ist:

$$E(H) = \Big(E(M_r^{HACK})-1\Big)\Big(T+E(W_G^{HACK})+E(X)\Big)+\tau_H+E(W_G^{HACK})+E(X). \quad (4.153)$$

$\tau_H$ ist die Nachrichtenverzögerung im Netzwerk bei einer hierarchischen Übertragungswiederholung. Da diese üblicherweise eine kürzere Distanz im Netzwerk überbrücken muss, wird in den numerischen Auswertungen $\tau_H$ auf einen kleineren Wert als $\tau$ gesetzt.

Um die gesamte Auslieferungsverzögerung zu bestimmen wird berücksichtigt, dass ein Empfänger mit Wahrscheinlichkeit $1-q_D$ die initiale Übertragung empfängt. Mit der Wahrscheinlichkeit $q_D$ benötigt ein Empfänger eine Übertragungswiederholung. Mit der Wahrscheinlichkeit $1-q_D$ kann der

Vaterknoten die Übertragungswiederholung durchführen, da dieser die Übertragung bereits erfolgreich empfangen hat. Hat auch der Vaterknoten die Übertragung nicht erfolgreich empfangen, so muss eine weitere Hierarchiestufe in Richtung Kontrollbaumwurzel für die Übertragungswiederholung sorgen. Die Multiplikation aller Wahrscheinlichkeiten mit den Verzögerungszeiten liefert die Durchschnittsverzögerung. Für jede weitere Stufe muss dazu erneut die Zeit für eine hierarchische Übertragungswiederholung, $E(H)$, berücksichtigt werden. Die Durchschnittsverzögerung lässt sich berechnen zu:

$$E(S_{\phi}^{HACK}) = \Big[\sum_{i=o}^{\bar{h}-2} q_D^i(1-q_D)\Big(E(I)+iE(H)\Big)\Big] + q_D^{\bar{h}-1}\Big(E(W_S^{HACK})+E(X)+(\bar{h}-1)E(H)\Big). \quad (4.154)$$

Die Umlaufverzögerung umfasst in Protokollklasse K-HACK und K-HNAPP lediglich eine lokale Gruppe, da keine aggregierten Quittungen zur Verfügung stehen. Die Umlaufverzögerung beträgt:

$$\begin{aligned} E(S_{RTD}^{HACK}) = & \underbrace{\Big(E(M^{HACK})-1\Big)\Big(T+E(W_S^{HACK})+E(X)\Big)}_{\textit{Übertragung senden}} \\ & + \underbrace{E(X)+E(W_S^{HACK})+\tau_H+E(W_G^{HACK})+E(X)}_{\textit{letzte, erfolgreiche Übertragung empfangen}} \\ & + \underbrace{E(Y)+E(W_G^{HACK})+\tau_H+E(W_S^{HACK})+E(Y).}_{\textit{letzte Quittung senden und empfangen}} \end{aligned} \quad (4.155)$$

Schließlich ist die Maximalverzögerung, um mit der Wahrscheinlichkeit $\gamma$ alle Nachrichten erfolgreich zuzustellen:

$$\gamma = 1-q_D^{M_\gamma^{HACK}} \quad (4.156)$$

$$M_\gamma^{HACK} = \frac{ln(1-\gamma)}{ln(q_D)}, \qquad \gamma \geq q_D \quad (4.157)$$

$$E(H) = \Big(E(M_\gamma^{HACK})-1\Big)\Big(T+E(W_G^{HACK})+E(X)\Big)+\tau_H+E(W_G^{HACK})+E(X) \quad (4.158)$$

$$E(S_\gamma^{HACK}) = E(W_S^{HACK})+E(X)+(\bar{h}-1)E(H). \quad (4.159)$$

In Formel 4.159 wird angenommen, dass Übertragungswiederholungen alle Knoten auf dem Pfad vom Sender bis zu einem zufälligen Empfänger $r$ betreffen, d.h. dass zuerst alle transitiven Vaterknoten von $r$ die Übertragung erfolgreich empfangen müssen, bevor eine Übertragung bzw. Übertragungswiederholung mit der geforderten Wahrscheinlichkeit $\gamma$ vom Empfänger $r$ empfangen werden kann, was einer oberen Schranke gleichkommt.

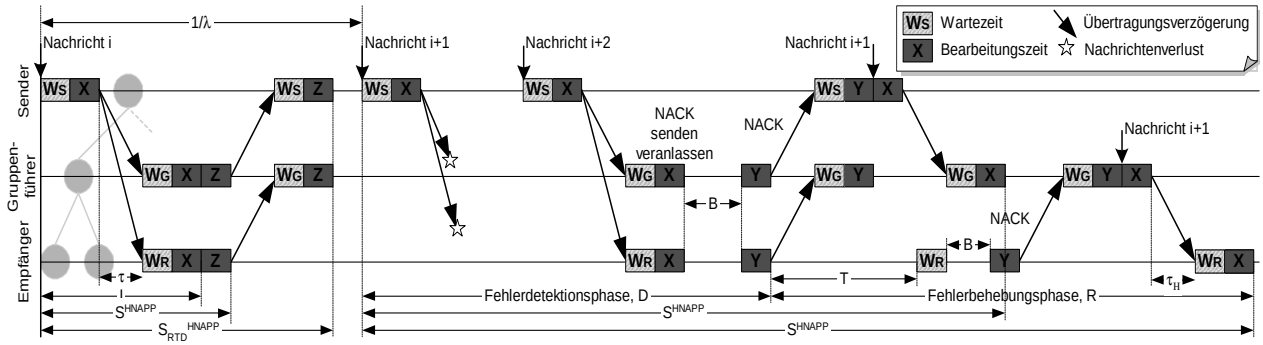



*Abbildung 4.12: Verzögerungskomponenten der Protokollklasse K-HNAPP*

#### 4.5.4.5 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-HNAPP)

**Mittlere Wartezeit einer Nachricht beim Sender:** Die in Abbildung 4.12 illustrierten Verzögerungskomponenten legen die Fehlerbehebung mit Hilfe von Multicast-Quittungen dar. Der Sender muss die folgenden vier Nachrichtenflüsse bearbeiten:

1. Der erste Nachrichtenfluss ist die initiale Datenübertragung $\lambda_t^S$ mit Rate $\lambda$ und Bearbeitungszeit $X$.

2. Der Fluss der ankommenden negativen Quittungen, die eine Übertragungswiederholung auslösen, wird mit $\lambda_r^S$ bezeichnet. Diese haben die Rate $\lambda\Big(E(M^{HNAPP})-1\Big)$ und Bearbeitungszeit $X+Y$.

3. Alle übrigen negativen Quittungen, die keine Übertragungswiederholung auslösen, bilden den Nachrichtenfluss $\lambda_n^S$ mit der Rate $\lambda E(\widetilde{L}^{HNAPP})-\lambda_r^S$. Die Rate berücksichtigt sowohl die zusätzlichen Quittungen der eigenen lokalen Gruppe als auch die empfangenen Quittungen anderer lokaler Gruppen. Die Bearbeitungszeit ist $Y$.

4. Schließlich den Fluss der periodischen Quittungen $\lambda_s^S$ mit der Rate $\lambda B(1-p_A)$ und der Bearbeitungszeit $Z$. $Z$ wird in den numerischen Auswertungen auf einen entsprechend kleinen Betrag gesetzt, um zu berücksichtigen, dass periodische Quittungen nicht nach jeder Datenübertragung gesendet werden.

Die Summe der Nachrichtenflüsse und ihrer Bearbeitungszeiten ergibt die Last des Senders:

$$\rho_S^{HNAPP} = \lambda E(M^{HNAPP})E(X)+\lambda E(\widetilde{L}^{HNAPP})E(Y)+\lambda B(1-p_A)E(Z). \quad (4.160)$$

Die mittlere Wartezeit einer Nachricht bis zur Bearbeitung ist folglich:

$$E(W_S^{HNAPP}) = \frac{\lambda_t^S E(X^2) + \lambda_r^S \Big( E(X^2) + E(Y^2) + 2E(X)E(Y) \Big) + \lambda_n^S E(Y^2) + \lambda_s^S E(Z^2)}{2(1-\rho_S^{HNAPP})}. \quad (4.161)$$

**Mittlere Wartezeit einer Nachricht beim Empfänger:** Auch ein Empfänger hat vier Nachrichtenflüsse zu bearbeiten:

1. $\lambda_t^R$ ist der Nachrichtenfluss empfangener Übertragungen der eigenen oder fremder lokaler Gruppen. Die Rate lässt sich aus der Bandbreitenanalyse ableiten und beträgt $\lambda \Big[ E(M^{HNAPP})(1 - q_D) + (G-1) p_l \Big( E(M^{HNAPP}) - 1 \Big)(1-q_D) \Big]$. Die Bearbeitungszeit ist $X$.
2. Der zweite Nachrichtenfluss $\lambda_{n,g}^R$ besteht aus den gesendeten negativen Quittungen mit der Rate $\lambda \Big( E(O_r^{HNAPP}) - 1 \Big) E(L_\phi^{HNAPP}) / E(R_N)$ und der Bearbeitungszeit $Y$.
3. Ein weiterer Nachrichtenfluss ergibt sich durch die empfangenen negativen Quittungen von anderen erfolglosen Empfängern $\lambda_{n,r}^R$. Dieser Nachrichtenfluss hat die aufwändige Rate bestehend aus den empfangenen Nachrichten von Empfängern der eigenen lokalen Gruppe $\lambda(1-q_N) \Big[ E(L_\phi^{HNAPP}) \Big( E(O^{HNAPP}) - 1 \Big) - E(L_\phi^{HNAPP}) / E(R_N) \Big( E(O_r^{HNAPP}) - 1 \Big) \Big]$ und fremder lokaler Gruppen $\lambda(G-1) p_l [E(O^{HNAPP}) - 1] E(L_\phi^{HNAPP})$. Nähere Ausführungen zur Berechnung der Rate können aus Abschnitt 4.5.2 entnommen werden. Die Bearbeitungszeit ist $Y$.
4. Der letzte Nachrichtenfluss $\lambda_s^R$ sind die gesendeten periodischen Quittungen mit der Rate $\lambda$ und der Bearbeitungszeit $Z$.

Die Gesamtanzahl der Übertragungsrunden $O^{HNAPP}$, die Anzahl der Übertragungsrunden für einen einzelnen Empfänger $O_r^{HNAPP}$ sowie die gesendeten negativen Quittungen $L_\phi^{HNAPP}$ und $R_N$ sind in Abschnitt 4.5.1.5 gegeben. Die Last des Empfängers ergibt sich aus den Nachrichtenflüssen zu:

$$\rho_R^{HNAPP} = \lambda_t^R + (\lambda_{n,g}^R + \lambda_{n,r}^R) E(Y) + \lambda_s^R E(Z). \quad (4.162)$$

Die mittlere Wartezeit einer Nachricht beim Empfänger ist:

$$E(W_R^{HNAPP}) = \frac{\lambda_t^R E(X^2) + (\lambda_{n,g}^R + \lambda_{n,r}^R) E(Y^2) + \lambda_s^R E(Z^2)}{2(1-\rho_R^{HNAPP})}. \quad (4.163)$$

**Mittlere Wartezeit einer Nachricht beim Gruppenführer:** Die Last und mittlere Wartezeit ergibt sich aus den Betrachtungen des Senders und des Empfängers:

$$\begin{aligned} \rho_G^{HNAPP} &= \lambda \Big( E(M^{HNAPP}) - 1 \Big) E(X) + \lambda E(\widetilde{L}^{HNAPP}) E(Y) + \lambda_s^R E(Z) \\ &\quad + \lambda_t^R E(X) + (\lambda_{n,g}^R + \lambda_{n,r}^R) E(Y) + \lambda_s^R E(Z) \quad (4.164) \\ E(W_G^{HNAPP}) &= \frac{\lambda_r^S \Big( E(X^2) + E(Y^2) + 2E(X)E(Y) \Big) + \lambda_n^S E(Y^2) + \lambda_s^S E(Z^2)}{2(1-\rho_G^{HNAPP})} \\ &\quad + \frac{\lambda_t^R E(X^2) + (\lambda_{n,g}^R + \lambda_{n,r}^R) E(Y^2) + \lambda_s^R E(Z^2)}{2(1-\rho_G^{HNAPP})}. \quad (4.165) \end{aligned}$$

**Gesamtverzögerung:** Die Länge der Fehlerdetektionsphase sowie der Fehlerbehebungsphase folgt direkt aus den Betrachtungen von K-NAPP:

$$\begin{aligned} E(D^{HNAPP}) &= \sum_{k=0}^{B-1} \binom{B-1}{k} {q_D}^k (1-q_D)^{B-1-k} \frac{{q_D}^{k+1}}{1-{q_D}^{k+1}} \frac{1}{\lambda} \\ &\quad + E(W_S^{HNAPP}) + E(X) + \tau + E(W_G^{HNAPP}) + E(X) + E(B^{HNAPP}) + E(Y) \qquad (4.166) \\ E(R^{HNAPP}) &= \Big(E(M_r^{HNAPP}) + E(O_{e,r}^{HNAPP}) - 2\Big)\Big(T + E(W_G^{HNAPP}) + E(B^{HNAPP}) + E(Y)\Big) \\ &\quad + \tau_H + E(W_G^{HNAPP}) + E(Y) + E(X) + \tau_H + E(W_G^{HNAPP}) + E(X). \qquad (4.167) \end{aligned}$$

Zu beachten ist, dass hier eine vereinfachende pessimistische Annahme darin besteht, stets einen Gruppenführer mit dessen Wartezeit $E(W_G^{HNAPP})$ zu veranschlagen. Die Verzögerung der initialen Übertragung ist:

$$E(I) = E(W_S^{HNAPP}) + E(X) + \tau + E(W_G^{HNAPP}) + E(X). \qquad (4.168)$$

Die Ermittlung der Durchschnittsverzögerung folgt aus der Protokollklasse K-HACK:

$$\begin{aligned} E(S_\phi^{HNAPP}) &= (1-q_D)E(I) + \Big[\sum_{i=1}^{\bar{h}-2} {q_D}^i (1-q_D)\Big(E(I) + E(D^{HNAPP}) + iE(R^{HNAPP})\Big)\Big] \\ &\quad + {q_D}^{\bar{h}-1}\Big(E(D^{HNAPP}) + (\bar{h}-1)E(R^{HNAPP})\Big). \qquad (4.169) \end{aligned}$$

Für die Umlaufverzögerung soll die Zeitspanne ermittelt werden, bis die periodischen Quittungen beim Sender eingetroffen sind. Dabei wird angenommen, dass eine periodische Quittung direkt im Anschluss an die betrachtete Datenübertragung gesendet wird. Da für periodische Quittungen keine Übertragungswiederholungen vorgesehen sind, findet ein Nachrichtenverlust keine Beachtung. Die Umlaufverzögerung beträgt demnach:

$$\begin{aligned} E(S_{RTD}^{HNAPP}) &= (1-q_D)^B E(I) \underbrace{+\Big(1-(1-q_D)^B\Big)\Big(E(D^{HNAPP}) + E(R^{HNAPP})\Big)}_{\textit{Übertragung senden}} \\ &\quad + \underbrace{E(Y) + E(W_G^{HNAPP}) + \tau_H + E(W_S^{HNAPP}) + E(Y).}_{\textit{letzte periodische Quittung senden und empfangen}} \qquad (4.170) \end{aligned}$$

Zu beachten ist, dass die Bearbeitungszeit von normalen Quittungen $E(Y)$ anstatt periodischen Quittungen $E(Z)$ Verwendung findet, da die Situation betrachtet wird, in der tatsächlich eine periodische Quittung gesendet wird.

Die Maximalverzögerung $E(S_\gamma^{HNAPP})$ berechnet sich wie folgt:

$$\begin{aligned} E(R^{HNAPP}) &= \Big(E(M_\gamma^{HNAPP}) + E(O_{e,r}^{HNAPP}) - 2\Big)\Big(T + E(W_G^{HNAPP}) + E(B^{HNAPP}) + E(Y)\Big) \\ &\quad + \tau_H + E(W_G^{HNAPP}) + E(Y) + E(X) + \tau_H + E(W_G^{HNAPP}) + E(X) \qquad (4.171) \\ E(S_\gamma^{HNAPP}) &= E(W_S^{HNAPP}) + E(X) + E(D^{HNAPP}) + (\bar{h} - 1)E(R^{HNAPP}). \qquad (4.172) \end{aligned}$$

$M_\gamma^{HNAPP}$ wird analog zur Protokollklasse K-HACK berechnet.

Die zwei verbleibenden Protokollklassen K-HAACK und K-HANAPP werden ähnlich zu den bisher ausgeführten Klassen analysiert. Die Berücksichtigung der zusätzlichen aggregierten Quittungen ist aufgrund der Quantitätsberechnungen aus Abschnitt 4.5.1 problemlos möglich. Die vollständige Analyse befindet sich darum im Anhang.

### 4.5.5 Erweiterung des Systemmodells zur Berücksichtigung räumlich abhängiger Nachrichtenverluste

In der bisherigen Analyse wurde zur Wahrung der Allgemeingültigkeit davon abgesehen, abhängige Nachrichtenverluste zu berücksichtigen. Eine räumliche Abhängigkeit der Verluste ist jedoch in der Realität vielfach gegeben, da häufig Pfade zu den unterschiedlichen Empfängern über gemeinsame Router laufen. Wie in Abschnitt 4.4 bereits erläutert wurde, konnte durch Messungen im MBone eine signifikante räumliche Abhängigkeit allerdings lediglich auf der *ersten* Verbindung vom Sender zum ersten Router festgestellt werden.

Aufgrund dieser Messungen wird eine Erweiterung des Netzwerkmodells vorgenommen, das die gemessenen Gegebenheiten im MBone besser berücksichtigen kann, hingegen nicht mehr allgemein z.B. für lokale Netze oder Intranetzwerke, Gültigkeit besitzt. Erstmals wurde die hier beschriebene Erweiterung der Analysen in [Maihöfer & Rothermel 2001e, Maihöfer & Rothermel 2002] vorgestellt. Die Abbildung 4.13 zeigt das Modell des Netzwerks. Der Sender ist mit einer verlustbehafteten Verbindung an das restliche Netzwerk (Hauptnetz, engl. Backbone) angebunden. Ein auftretender Fehler auf dieser Verbindung wird von allen Empfängern erfahren. Das Hauptnetz selbst wird als verlustfrei angenommen, was bestätigt durch die Messungen in [Yajnik et al. 1996] und [Yajnik et al. 1999] weitestgehend der Realität im MBone entspricht. Schließlich ist jeder Empfänger über eine verlustbehaftete Verbindung an das Hauptnetz angebunden. Fehler auf diesen Verbindungen weisen keine räumlichen Abhängigkeiten zu Fehlern anderer Verbindungen auf. Die Annahmen dieses Modells entsprechen den Annahmen in früheren Arbeiten [Kasera et al. 1998, Nonnenmacher et al. 1998].

Die Ende-zu-Ende-Verlustwahrscheinlichkeit zwischen Sender und Empfänger soll weiterhin $q_D$ betragen. Diese Wahrscheinlichkeit wird zu gleichen Teilen aufgeteilt in eine Verlustwahrscheinlichkeit

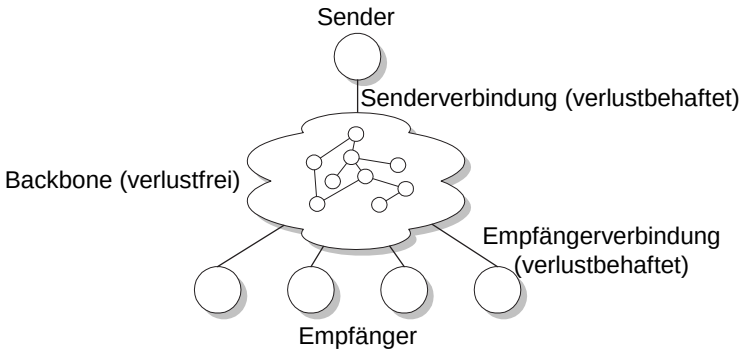



*Abbildung 4.13: Netzwerkmodell mit räumlich abhängigen Nachrichtenverlusten*

der Senderverbindung und der Empfängerverbindung, $q_{D'}$. Für die Berechnung ergibt sich die Bedingung $(1-q_{D'})(1-q_{D'}) = (1-q_D)$, da die Wahrscheinlichkeit für einen korrekten Empfang einer Nachricht nach der Senderverbindung $(1-q_{D'})$ und der Empfängerverbindung (ebenfalls $1-q_{D'}$) identisch mit der Wahrscheinlichkeit für einen korrekten Nachrichtenempfang beim Empfänger ist $(1-q_D)$. Die Verlustwahrscheinlichkeit der Sender– und Empfängerverbindung folgt daraus zu:

$$q_{D'} = 1-\sqrt{1-q_D}. \tag{4.173}$$

Die erwartete Anzahl notwendiger Übertragungen, um eine Datennachricht bei allen Empfängern erfolgreich zu empfangen, ist die Summe der Übertragungswiederholungen aufgrund von Nachrichtenverlust auf der Senderverbindung, $E(M_S)$, der Empfängerverbindung inklusive des Quittungsverlusts, $E(M_T)$, und der initialen Übertragung:

$$E(M) = E(M_S)+E(M_T)+1. \tag{4.174}$$

Die Anzahl an Übertragungswiederholungen aufgrund von Nachrichtenverlust auf der Senderverbindung beträgt:

$$E(M_S) = \frac{1}{1-q_{D'}}-1. \tag{4.175}$$

Wie der Formel zu entnehmen ist, werden diese analog zu den notwendigen Übertragungen für einen einzelnen Empfänger berechnet, der in diesem Fall das Hauptnetz darstellt (siehe Formel 4.4). Die Anzahl an Übertragungswiederholungen durch einen Verlust auf den Empfängerverbindungen werden ebenfalls mit der bisherigen Berechnungsmethode aus Formel 4.20 ermittelt. Für senderinitiierte Protokolle muss die Verlustwahrscheinlichkeit $q_D = (q_{D'}+(1-q_{D'})p_A)$ eingesetzt werden, während für empfängerinitiierte Protokolle $q_D = q_{D'}$ gilt. Am Beispiel von Protokollklasse K-HACK beträgt die Anzahl an Übertragungswiederholungen durch den Verlust auf den Empfängerverbindungen:

$$E(M_T^{HACK}) = \sum_{i=1}^{B} \binom{B}{i} (-1)^{i+1} \frac{1}{1-(q_D\prime + (1-q_D\prime)p_A)^i} - 1. \quad (4.176)$$

Neben den Übertragungswiederholungen muss zudem die Berechnung der Umlaufverzögerung für die empfängerinitiierten Protokolle dem geänderten Netzwerkmodell angepasst werden:

$$\begin{aligned} E(S_{RTD}^{HNAPP}) = &\left[1 - \left(q_D\prime + (1-q_D\prime)(1-(1-q_D\prime)^B)\right)\right] E(I) \\ &+ \left(q_D\prime + (1-q_D\prime)(1-(1-q_D\prime)^B)\right)\left(E(D^{HNAPP}) + E(R^{HNAPP})\right) \\ &+ E(Y) + E(W_G^{HNAPP}) + \tau_H + E(W_S^{HNAPP}) + E(Y), \end{aligned} \quad (4.177)$$

und analog für die restlichen empfängerinitiierten Protokollklassen K-NAPP und K-HANAPP. Die numerischen Ergebnisse im folgenden Abschnitt basieren auf diesem erweiterten Netzwerkmodell mit räumlich abhängigen Verlusten.

## 4.6 Numerische Ergebnisse der Analyse

### 4.6.1 Einführung

Zur Veranschaulichung der analytischen Berechnungen werden im Verlauf dieses Abschnitts numerische Ergebnisse in graphischer Form dargestellt. Bei der Berechnung der Ergebnisse mussten zwei wesentliche Kompromisse gegenüber den analytisch exakten Resultaten eingegangen werden. Der erste Kompromiss betrifft die unteren und oberen Grenzen der endlichen Summen. Für diese ergeben sich bei der numerischen Auswertung oftmals keine ganzzahligen Werte, was eine direkte Berechnung verhindert. Eine einfache Lösung ist die Rundung der Grenzen auf den nächsten ganzzahligen Wert. Nachteilig an dieser einfacheren Lösung sind die entstehenden Unstetigkeitsstellen an den Übergangspunkten zur nächstkleineren oder –größeren Grenze der Summe. Um diesem Problem auszuweichen und einen einfachen Vergleich mit den kontinuierlichen Ergebnissen einer Simulation zu ermöglichen, wurde folgende Lösung gewählt. Nicht ganzzahlige Grenzwerte wurden zwischen den benachbarten ganzzahligen Grenzwerten linear interpoliert. Die resultierenden Kurven sind folglich stetig.

Ein weiteres Problem betrifft die Berechnung der Anzahl an Übertragungswiederholungen. Die Formel dazu lautet (siehe Abschnitt 4.5.1):

$$E(M) = \sum_{i=1}^{R} \binom{R}{i} (-1)^{i+1} \frac{1}{1-\tilde{p}^i}.$$

Die Berechnung der Kombination $\binom{R}{i}$ hat sich als äußerst problematisch für große Empfängergruppen $R$ herausgestellt. $\binom{R}{i}$ wird mit Hilfe der Formel $\frac{R!}{i!(R-i)!}$ ermittelt. Die Berechnung der Fakultät beträgt bereits für 100 Empfänger $100! \approx 10^{157}$, so dass die Berechnung einer größeren Empfängergruppe nicht möglich ist. Für die Berechnung der Anzahl an Übertragungswiederholungen für eine Gruppengröße von mehr als 35 Empfängern wurde auf ein Näherungsverfahren ausgewichen, das in [Pingali et al. 1994] beschrieben ist und dort ebenfalls für die numerischen Auswertungen eingesetzt wird:

$$\begin{aligned} E(M) &= z_1 + \frac{z_2 - z_3}{log_{\tilde{p}}} \\ z_1 &= \sum_{i=1}^{35} \binom{35}{i} (-1)^{i+1} \frac{1}{1-\tilde{p}^i} \\ z_2 &= \sum_{i=1}^{35} \frac{1}{i} \\ z_3 &= \sum_{i=1}^{R} \frac{1}{i}. \end{aligned}$$

Die folgenden Abschnitte zeigen die Ergebnisse der numerischen Auswertung. Dabei wurde stets das verbesserte Systemmodell mit räumlich abhängigen Verlusten aus dem vorigen Abschnitt zugrundegelegt. Dieses führt im Vergleich zum Modell mit unabhängigen Verlusten zu einer verringerten Anzahl notwendiger Nachrichtenwiederholungen. Typischerweise lag die Verringerung bei ca. 10–20%. Durch Vergleiche mit Simulationsergebnissen wurde festgestellt, dass das Systemmodell mit räumlich abhängigen Verlusten tatsächlich die Realität in den simulierten Netzen exakter widerspiegelt.

### 4.6.2 Numerische Ergebnisse zur Bandbreite

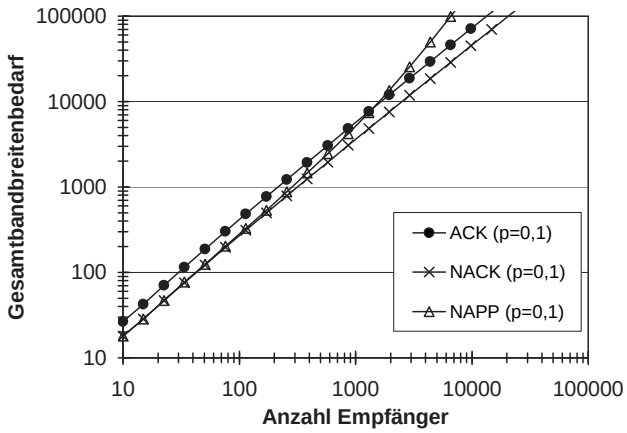
Die ersten dargestellten Ergebnisse zeigen den Bandbreitenbedarf der untersuchten Protokollklassen, zum Einen als Gesamtbandbreitenbedarf aller Kommunikationskanäle auf Transportschicht und zum Anderen als Bandbreitenbedarf für ein individuelles Mitglied der Multicast-Gruppe. Um unabhängig von spezifischen Wertebelegungen zu sein, werden normierte Ergebnisse dargestellt. Dazu wurde der Bandbreitenbedarf für eine Datennachricht auf eins normiert. Der Bandbreitenbedarf für eine Kontrollnachricht ist üblicherweise deutlich kleiner.[2] Hier wurde deshalb 0,1 als Bandbreitenbedarf für eine Quittung angesetzt. Für die periodischen Quittungen der Protokollklasse K-HNAPP wurde angenommen, dass jeweils auf zehn Datennachrichten eine periodische Quittung versendet wird. Deshalb

[2] Messungen im Internet [Gramsamer 2001] haben ergeben, dass die durchschnittliche Paketgröße bimodal verteilt ist. Danach haben kleine Pakete eine durchschnittliche Größe von 44 Byte und große Pakete von 564 Byte. Kleine Pakete sind typischerweise die Kontrollpakete von TCP und damit in ihrer Funktion den hier verwendeten Quittungen sehr ähnlich. Die Größe von Quittungen auf ein Zehntel der Größe von Datenpakete zu setzen, scheint demnach angemessen zu sein.

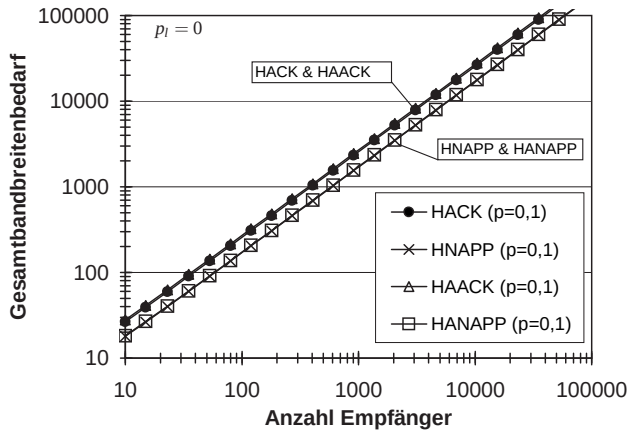
ist deren Bandbreitenbedarf lediglich 0,01 (vergleiche auch [Levine & Garcia-Luna-Aceves 1998]). Der normierte Bandbreitenbedarf kann auf reale Gegebenheiten skaliert werden. Wird beispielsweise eine durchschnittliche Größe der Datennachrichten von 1000 Byte angenommen, spiegeln die Ergebnisse unmittelbar den Bandbreitenbedarf in KByte wider.

Zur Interpretation von Abbildung 4.14 sind weitere Variablenbelegungen von Interesse. Die Nachrichtenverlustwahrscheinlichkeit eines Kommunikationskanals auf Transportschicht beträgt 10%, da durch Messungen im MBone dies als realistisch angesehen werden kann [Yajnik et al. 1996]. Zum Vergleich werden zudem Ergebnisse mit 1% Nachrichtenverlustwahrscheinlichkeit präsentiert, die bei einer zuverlässigeren Funktionsweise der Router in Zukunft realistischer sein dürfte. Die Wahrscheinlichkeit für simultanes Senden von negativen Quittungen wurde auf 20% gesetzt. Dies begründet sich durch folgende Überlegung. Bei einem Protokoll mit Quittungsunterdrückung wird eine zufällig gewählte Zeit auf eintreffende Quittungen gewartet, bevor gegebenenfalls die eigene negative Quittung gesendet wird. Die zufällige Wartezeit sollte dabei in einem Bereich liegen, der deutlich größer ist als die durchschnittliche Nachrichtenverzögerung im Netzwerk, um wirkungsvoll zu arbeiten. Sie sollte aber auch nicht zu groß sein, um die Verzögerung für Übertragungswiederholungen klein zu halten. In [Yamamoto et al. 1997] wird empfohlen, den zehnfachen Wert der Nachrichtenverzögerung im Netzwerk zu wählen. In diesen Auswertungen wurde der fünffache Wert gewählt, da lokale Gruppen in hierarchischen Protokollen kleiner sind und dies einen guten Kompromiss zwischen Nachrichtenanzahl und Verzögerung für Übertragungswiederholungen darstellt. Dadurch ist die Wahrscheinlichkeit für gleichzeitiges Senden von negativen Quittungen 1/5, d.h. 20%. Für die hierarchischen Protokolle wurde ein Verzweigungsgrad von 10 Kindknoten pro Gruppenführer gewählt und vorerst eine Überlappungswahrscheinlichkeit lokaler Gruppen von $p_l = 0$.

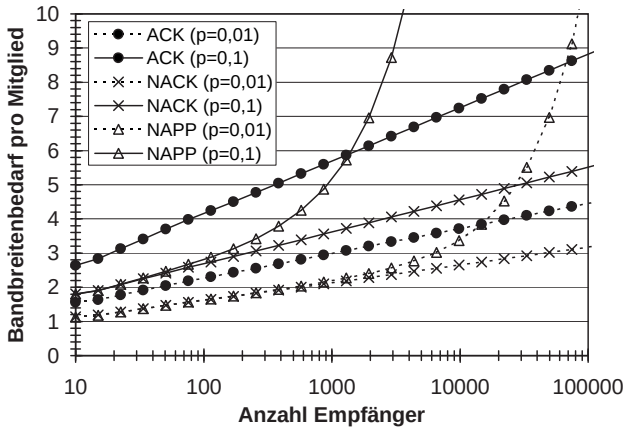
Abbildungen 4.14a und 4.14b zeigen den summierten Gesamtbandbreitenbedarf auf Transportschicht für den Sender und alle Empfänger flacher und hierarchischer Protokolle entsprechend der Berechnung von $E(W)$ in Abschnitt 4.5.2. Die gestrichelten Kurven legen eine Nachrichtenverlustwahrscheinlichkeit von 1% zugrunde ($p = 0,01$) und die durchgezogenen Kurven von 10% ($p = 0,1$). Zu beachten ist, dass beide Achsen logarithmisch skaliert sind. Leider lassen sich dadurch nur schwer Unterschiede ausmachen, da das überproportionale Wachstum der flachen Protokollklassen kaum ersichtlich ist. In Abbildung 4.14c und 4.14d wurde deshalb der Bandbreitenbedarf pro beteiligtem Mitglied dargestellt. In Abbildung 4.14d wurde zur Verbesserung der Übersicht auf die Darstellung von K-HANAPP verzichtet. Der Bandbreitenbedarf von K-HANAPP ist fast identisch mit dem von K-HNAPP. Für alle hierarchischen Protokolle zeigt sich, dass deren Bandbreitenbedarf pro Mitglied unabhängig von der Mitgliederzahl ist, während für die flachen Protokolle der Bandbreitenbedarf pro Mitglied mit der Mitgliederzahl zunimmt. Vor allem die Klasse K-NAPP zeigt ein ungünstiges Verhalten, verursacht durch die hohe Anzahl an Multicast-Quittungen bei einer großen Mitgliederzahl. Der Bandbreitenbedarf von K-ACK und K-NACK wächst dagegen lediglich linear, da einzig die Anzahl an Übertragungswiederholungen zunimmt, die ein Mitglied zu empfangen hat, jedoch nicht die



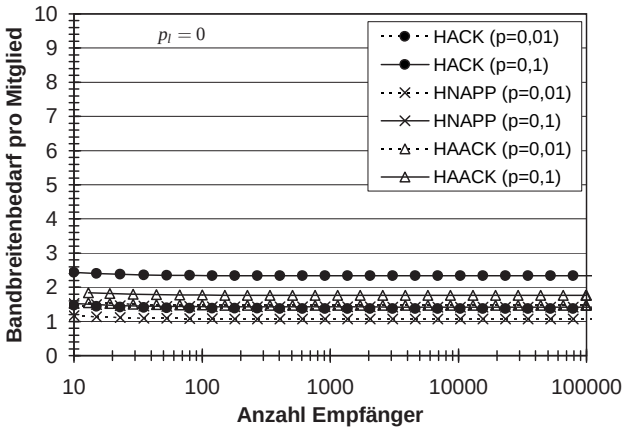

(a) Bandbreitenbedarf flacher Protokolle




(b) Bandbreitenbedarf hierarchischer Protokolle




(c) Bandbreitenbedarf pro Mitglied flacher Protok.




(d) Bandbreitenbedarf pro Mitglied hierarch. Protok.

*Abbildung 4.14: Gesamtbandbreitenbedarf einer Multicast-Gruppe und Bandbreitenbedarf pro Mitglied auf Transportschicht*

Anzahl negativer Quittungen anderer Mitglieder. Für hierarchische Protokolle bleiben beide Größen konstant, da diese durch den Verzweigungsgrad und nicht die Gruppengröße bestimmt werden. Dies trifft zu, solange es keine Lokalgruppenüberlappung gibt.

Die Abbildung 4.15 zeigt die Ergebnisse für Protokollklasse K-HNAPP unter Berücksichtigung von Lokalgruppenüberlappung mit einer Nachrichtenverlustwahrscheinlichkeit von 10%. Teil a) der Abbildung zeigt den Bandbreitenbedarf pro Mitglied mit der Wahrscheinlichkeit $p_l$ für eine Lokalgruppenüberlappung von 0,001. Die generelle Tendenz ist, dass der Aufwand pro Mitglied mit der Gruppengröße wächst. Der Einfluss des Verzweigungsgrads muss differenzierter betrachtet werden. Bei kleinen Gruppen führt ein kleiner Verzweigungsgrad zu den besten Ergebnissen, da hier die Lokalgruppenüberlappung keinen bedeutenden Einfluss hat und mit kleinen lokalen Gruppen im Mit-

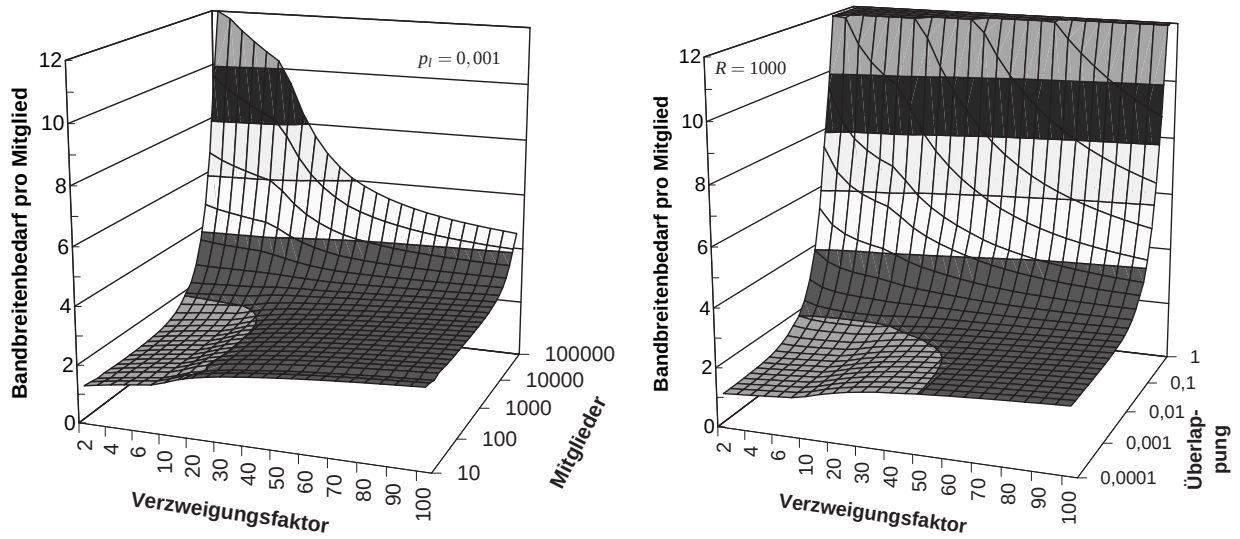
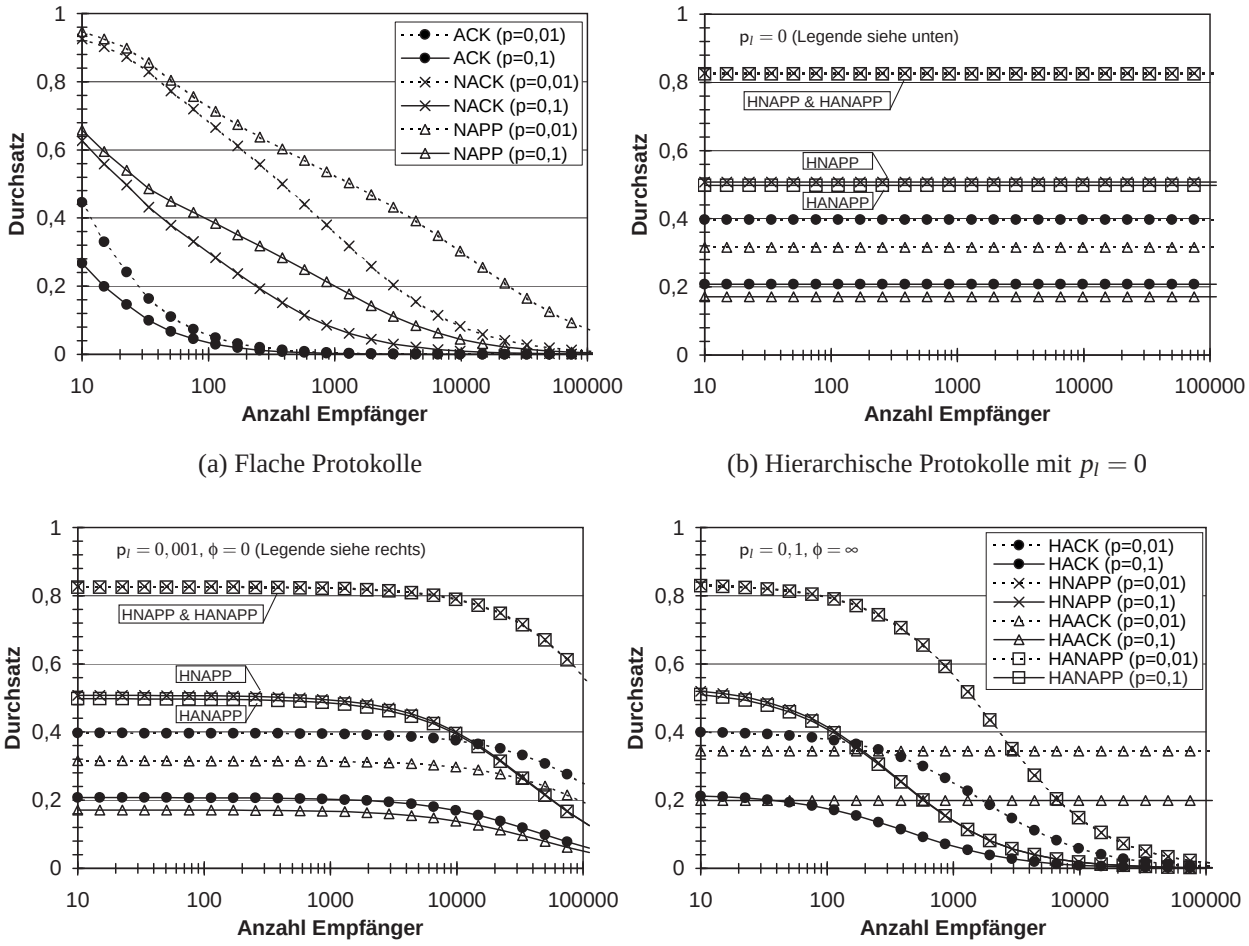



*Abbildung 4.15: Einfluss der Mitgliederzahl, der Lokalgruppenüberlappung und des Verzweigungsgrads auf den Bandbreitenbedarf pro Mitglied des Protokolls K-HNAPP*

tel weniger Übertragungswiederholungen notwendig sind. Ganz anders ist die Situation bei großen Gruppen. Bei kleinem Verzweigungsgrad und demnach vielen lokalen Gruppen führt die Lokalgruppenüberlappung zu einem hohen Aufwand bei allen lokalen Gruppen, bedingt durch den Empfang von Nachrichten anderer lokaler Gruppen. In dieser Situation bringt es deutliche Vorteile, die Anzahl lokaler Gruppen durch einen größeren Verzweigungsgrad zu beschränken. Abbildung 4.15b zeigt die Abhängigkeit von der Lokalgruppenüberlappung bei 1000 Empfängern. Wiederum zeigt sich, dass bei geringer Wahrscheinlichkeit für Lokalgruppenüberlappung, in diesem Beispiel bis etwa 0,01, ein kleiner Verzweigungsgrad die besten Ergebnisse liefert, während bei größerer Wahrscheinlichkeit für Lokalgruppenüberlappung auch ein größerer Verzweigungsgrad konfiguriert werden sollte.

Abbildung 4.16 zeigt den bandbreitenbeschränkten Durchsatz der untersuchten Protokolle. Durch den normierten Bandbreitenbedarf ergibt sich auch ein normierter Durchsatz, dessen Maximalwert eins beträgt. Der maximale Durchsatz von eins könnte dann erreicht werden, wenn der Sender lediglich eine Datennachricht senden müsste und jeder Empfänger lediglich eine Datennachricht empfangen müsste und keine Notwendigkeit für das Senden oder Empfangen von Quittungen bestehen würde. Der normierte Durchsatz kann auf reale Gegebenheiten skaliert werden. Angenommen, alle Mitglieder der Multicast-Gruppe sind über eine 64 KBit/s ISDN Verbindung mit dem Internet verbunden und diese Anbindung würde jeweils den Flaschenhals darstellen, dann entspricht der normierte Maximaldurchsatz von eins dem Durchsatz dieses Szenarios von 64 KBit/s.

Aus Abbildung 4.16 wird ersichtlich, dass der Durchsatz flacher Protokolle für große Empfängergruppen einbricht. Vor allem für die Nachrichtenverlustwahrscheinlichkeit von 10% erbringt selbst




(a) Flache Protokolle

(b) Hierarchische Protokolle mit $p_l = 0$

(c) Hierarchische Protokolle mit $p_l = 0,001$

(d) Hierarchische Protokolle mit $p_l = 0,1, \phi = \infty$

*Abbildung 4.16: Bandbreitenbeschränkter Durchsatz*

die beste flache Protokollklasse K-NAPP bei 100000 Empfängern nur 0,4% des theoretischen Maximaldurchsatzes, da die Quittungsbearbeitungen und die Übertragungswiederholungen kaum mehr das Senden neuer Nachrichten erlauben. Für das Beispiel der 64 KBit/s ISDN Verbindung verbleibt damit noch 256 Bit/s realer Nutzdatendurchsatz. Signifikant besser schneiden die hierarchischen Protokolle ab. Die besten Klassen K-HNAPP und K-HANAPP können bei 1% Nachrichtenverlustwahrscheinlichkeit 80% der Bandbreite für den Nutzdatendurchsatz aufwenden und selbst bei 10% Nachrichtenverlustwahrscheinlichkeit immer noch 50%. Ohne Lokalgruppenüberlappung ist dieses Resultat unabhängig von der Gesamtgruppengröße aufgrund der konstanten Größe lokaler Gruppen.

Mit Lokalgruppenüberlappung sind die Ergebnisse für hierarchische Protokolle schlechter und zudem abhängig von der Gruppengröße, wie Abbildung 4.16c-d veranschaulicht. Für $p_l = 0,001$ liegt der Durchsatz aller hierarchischen Protokolle noch deutlich über dem der flachen Protokolle. Für

$p_l = 0,1$ ist der Durchsatz von K-HACK auf dem Niveau von K-NAPP und auch K-HNAPP sowie K-HANAPP erzielen trotz eines doppelt so hohen Durchsatzes kein überzeugendes Ergebnis. Dem sind allerdings die folgenden Bemerkungen hinzuzufügen. Erstens ist eine Lokalgruppenüberlappung von 10% bereits ein hoher Wert, der in der Praxis deutlich unterschritten werden dürfte.[3] Zweitens zeigt das Beispiel für die Protokollklasse K-HAACK in Abbildung 4.16d einen überzeugenden Durchsatz unabhängig von der Lokalgruppenüberlappung und der Gruppengröße. Hier wurden die Übertragungswiederholungen derart konfiguriert, dass prinzipiell Unicast verwendet wird ($\phi = \infty$), was bei einer hohen Lokalgruppenüberlappung die sinnvollste Entscheidung darstellt. Schließlich beziehen sich die gezeigten Ergebnisse auf einen Verzweigungsgrad von 10 Kindknoten pro Gruppenführer. Bereits die Ergebnisse zum Gesamtbandbreitenbedarf haben dargelegt, dass ein größerer Verzweigungsgrad bessere Ergebnisse liefern kann. In Abbildung 4.17 wird dies am Beispiel der Protokollklasse K-HNAPP dargelegt. Angenommen wurde hier eine Nachrichtenverlustwahrscheinlichkeit von 10%, eine Lokalgruppenüberlappung von 0,001 auf der linken Seite der Abbildung und 1000 Empfänger auf der rechten Seite der Abbildung. Mit einer Lokalgruppenüberlappung von 0,001 kann mit einem Verzweigungsgrad von 100 noch 13% des maximalen Durchsatzes bei 100000 Empfängern erreicht werden. Im Vergleich dazu erreicht die beste flache Protokollklasse K-NAPP nur 0,4%.

### 4.6.3 Numerische Ergebnisse zur CPU-Belastung

Die Ergebnisse zur CPU-Belastung zeigen ähnliche Ergebnisse wie die bereits besprochenen Ergebnisse zum Bandbreitenbedarf. Abbildung 4.18 zeigt den CPU-beschränkten Durchsatz für flache und hierarchische Protokolle. Auch hier wurde der Durchsatz normiert und deshalb folgende Wertebelegungen angenommen: $E(X) = E(Y) = 1$ für alle Daten, Quittungen und Zeitgeber bis auf die für periodische Quittungen, die mit $0,1$ belegt wurden. Der CPU-Aufwand für die Quittungsbearbeitung ist im Gegensatz zur Bandbreite nicht geringer, da die Nachrichtengröße dafür eine untergeordnete Rolle spielt. Der Hauptaufwand wird durch die Unterbrechungsbehandlung verursacht (vergleiche auch [Pingali et al. 1994]). Das Ergebnis für hierarchische Protokolle basiert auf einer Lokalgruppenüberlappung von 0,001.

Generell gilt für den CPU-beschränkten Durchsatz, dass ein geringerer Durchsatz verglichen mit dem theoretischen Maximaldurchsatz von eins erzielt wird als bei der Bandbreitenanalyse. Durch den gleich großen CPU-Aufwand für die Nachrichten– *und* die Quittungsbearbeitung und durch die

[3] In der Literatur lassen sich keine Messergebnisse zu realistischen Lokalgruppenüberlappungen finden. Die dafür notwendigen großen Teilnehmerzahlen an einer zuverlässigen Multicast-Übertragung dürften gegenwärtig das Haupthindernis darstellen. Mit Simulationen konnten diese Daten ebenfalls nicht erhoben werden, da die notwendigen großen Netze und großen Teilnehmerzahlen nicht realisiert werden konnten. Werden hingegeben kleine Netze simuliert, so ergeben sich unrealistisch dichte Multicast-Gruppen, die eine erhöhte Lokalgruppenüberlappung aufweisen (siehe auch Abschnitt 4.7).

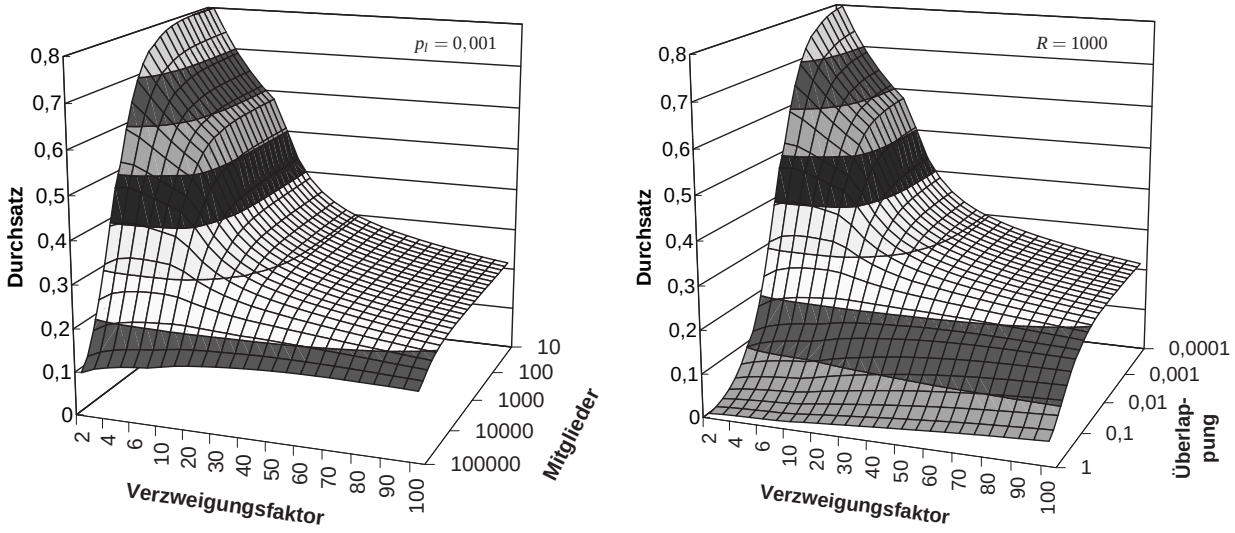



*Abbildung 4.17: Einfluss der Mitgliederzahl, der Lokalgruppenüberlappung und des Verzweigungsgrads auf den Durchsatz des hierarchischen Protokolls K-HNAPP*

Berücksichtigung der Zeitgeberbearbeitung zusätzlich zum Versenden und Empfangen von Nachrichten kommt dieses Ergebnis zustande. Der normierte theoretische Maximaldurchsatz von eins würde erreicht werden, wenn lediglich eine Datennachricht vom Sender und jeweils eine Datennachricht von jedem Empfänger bearbeitet werden müsste. Da der Sender eines hierarchischen senderinitiierten Protokolls der Klasse K-HACK bei einem Verzweigungsgrad von zehn zusätzlich zehn Quittungen pro Datennachricht bearbeiten muss, ist der normierte Durchsatz bereits maximal 1/11, Übertragungswiederholungen und Zeitgeberbearbeitung noch nicht eingerechnet.

Bedeutungsvoller als die absoluten Werte sind die Tendenzen für unterschiedliche Nachrichtenverlustwahrscheinlichkeiten und Mitgliederzahlen. Diese Tendenzen sind ähnlich zur Bandbreitenanalyse, wenn sie auch hier verstärkt auftreten. Demnach ist der Durchsatzabfall flacher Protokollklassen mit steigender Gruppengröße stärker ausgeprägt als bei der Bandbreitenanalyse, wodurch die Notwendigkeit hierarchische Verfahren für große Teilnehmerzahlen einzusetzen noch deutlicher wird. Beispielsweise ergibt sich für K-NAPP zwischen 10 und 100000 Empfängern ein Durchsatzabfall um mehr als den Faktor 600, welcher bei der Bandbreitenanalyse lediglich etwas mehr als den Faktor 100 betragen hat. Auch dies ist durch die zusätzlich zu bearbeitenden Zeitgeber verursacht. Für hierarchische Protokollklassen ergeben sich die gleichen Vorteile bei einer großen Gruppengröße, aber auch die selben Abhängigkeiten von der Lokalgruppenüberlappung.

Das folgende Zahlenbeispiel verdeutlicht die Ergebnisse. Da der Durchsatz durch den schwächsten Prozessor aller Multicast-Teilnehmer beschränkt wird, soll ein 100 MHz Prozessor angenommen werden. Dieser Prozessor soll 50% seiner Zeit für die Multicast-Übertragung aufwenden, da die restliche

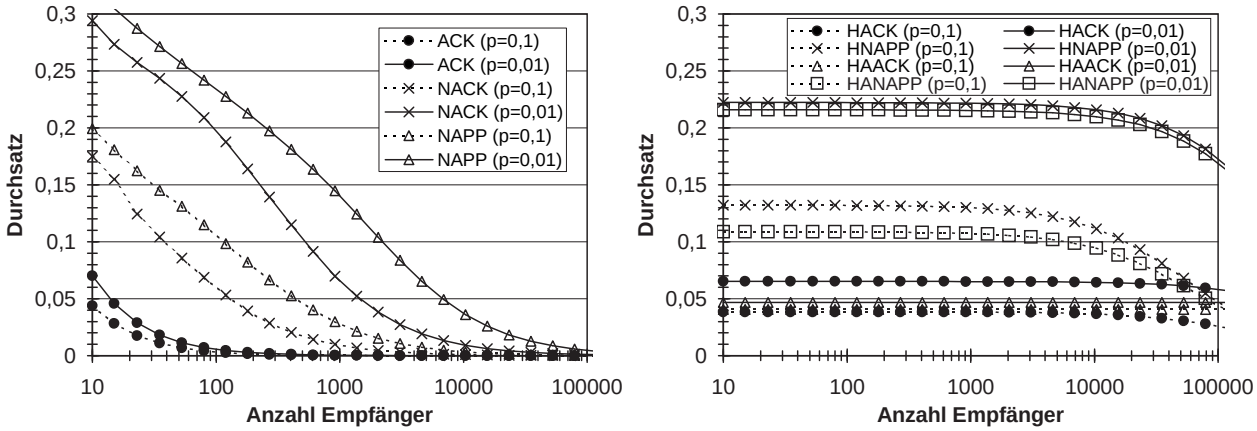



*Abbildung 4.18: CPU-beschränkter Durchsatz mit Lokalgruppenüberlappung $p_l = 0,001$*

Zeit für das Betriebssystem und andere Anwendungen benötigt wird. Desweiteren soll angenommen werden, dass pro 1000 Taktzyklen jeweils eine Datennachricht mit der Größe 1 KB versendet werden kann. Damit kommt dieser Rechner auf eine theoretische maximale Datenübertragungsrate von 50 MB pro Sekunde.[4] Bei Protokollklasse K-NAPP kann mit 100000 Empfängern lediglich 0,03% des maximalen Durchsatzes erzielt werden, also 15 KB pro Sekunde. Im Gegensatz dazu erzielt K-HNAPP mit einer Lokalgruppenüberlappung von 0,001 noch 4% des Maximaldurchsatzes, also 2 MB pro Sekunde.

### 4.6.4 Numerische Ergebnisse zur Verzögerung

Für die numerischen Auswertungen der Verzögerung müssen eine Reihe von Variablen mit praxisnahen Werten belegt werden, die nachfolgend kurz erläutert werden. Für die Nachrichtenverzögerung im Netzwerk $\tau$ wurde 100*ms* angenommen. In [Carter & Crovella 1996] und [Theilmann 2000] wurden dazu umfangreiche Messungen durchgeführt und durchschnittliche Umlaufzeiten im Internet von 241 und 256*ms* ermittelt. Der Median lag bei 125 bzw. 186*ms*. Der hier gewählte Wert von 100*ms* für die einfache Strecke liegt ungefähr zwischen diesen gemessenen Werten. Die Zeitgeber sollten zumindest auf den doppelten Wert der Nachrichtenverzögerung $\tau$ im Netzwerk eingestellt werden, damit genügend Zeit für die Daten in die eine Richtung und die Quittungen in die Rückrichtung bleibt. In den Auswertungen wurde mit 300*ms* der dreifache Wert gewählt, damit nicht unnötig oft fälschlicherweise Nachrichtenverlust detektiert wird. Entsprechend den Messungen von Kasera et al. [Kasera et al. 1997] über typische Nachrichtenbearbeitungszeiten, wurde für die Bearbeitung von Datennachrichten

[4]Dieser Wert ist tatsächlich ein realistischer Maximalwert, wenn er auch meistens nicht durch die reine CPU-Leisung, sondern maßgeblich durch die Bandbreite des Rechnerbuses begrenzt wird (siehe beispielsweise [Ahlers 2001]).

$X = 500\mu s$ und für die Bearbeitung von Quittungen $Y = 100\mu s$ angenommen. Die Zeit für die zufällige Verzögerung der negativen Quittung bei K-NAPP, K-HNAPP und K-HANAPP wurde auf den fünffachen Wert der Nachrichtenverzögerung im Netzwerk gesetzt (siehe Begründung in Abschnitt 4.6.2 und [Yamamoto et al. 1997]).

Bisher wurde dem Problem der Berechnung der zweiten Momente $E(X^2)$ und analog für $Y$, wie es beispielsweise in Formel 4.106 und vielen weiteren ebenfalls auftritt, keine Beachtung geschenkt. Die Berechnung erfolgt über die Gleichung $Var(X) = E(X^2) - (E(X))^2 \rightarrow E(X^2) = Var(X) + (E(X))^2$ [Rüegg 1986]. Die Berechnung des Quadrats des Erwartungswerts ist ausgehend von obigen Wertannahmen problemlos möglich. Für die Berechnung der Varianz muss allerdings eine Wahrscheinlichkeitsverteilung angenommen werden, da die Ergebnisse in [Kasera et al. 1997] darüber keine Aussage treffen. Die einfachste Annahme ist, dass die Nachrichtenbearbeitungszeiten konstant sind, d.h. keine Varianz besitzen. Diese Annahme wurde bereits in [Yamamoto et al. 1997] getroffen und wird auch hier für die Berechnung der numerischen Ergebnisse zugrundegelegt. Durch die Annahme von $Var(X) = 0$ ergibt sich für das zweite Moment $E(X^2) = (E(X))^2$. Die Überprüfung mit anderen Wahrscheinlichkeitsverteilungen ergab keine wesentlichen Abweichungen in den Resultaten, da die Verzögerungen hauptsächlich durch die Nachrichtenverzögerung im Netzwerk und die Zeitgeber für Übertragungswiederholungen bestimmt werden. Folglich ist diese Annahme als unkritisch anzusehen.

In Abbildung 4.19 sind die Durchschnittsverzögerungen aller untersuchten Protokollklassen dargestellt. Die gestrichelten Kurven legen eine Nachrichtenverlustwahrscheinlichkeit von 10% zugrunde und die durchgezogenen Kurven von 1%. Die Senderate beträgt für alle Ergebnisse $0,1\ 1/ms$. Auffällig sind bei den flachen Protokollklassen die steil ansteigenden Verzögerungen beim Überschreiten bestimmter Teilnehmerzahlen. Diese kennzeichnen die Schranke, ab der die betrachtete Protokollklasse bedingt durch die Überlastung eines Gruppenmitglieds nicht mehr mit der Teilnehmerzahl skaliert. Für eine Nachrichtenverlustwahrscheinlichkeit von 10% liegt diese Schranke vor allem für K-ACK ausgesprochen niedrig bei 30 Gruppenmitgliedern. Folgende überschlagende Betrachtung lässt dies als realistisch erscheinen. Der Sender benötigt für das Senden der Datennachricht $0,5ms$ und für das Empfangen von 30 Quittungen jeweils $0,1ms$, somit insgesamt $3,5ms$. Für jede Übertragungswiederholung fällt der gleiche Aufwand nochmals an. Bei 30 Empfängern und einer Nachrichtenverlustwahrscheinlichkeit von 10% werden entsprechend Formel 4.20 ungefähr drei Übertragungen benötigt, was schließlich zu einem Aufwand von ungefähr 10ms beim Sender führt. Dies entspricht bei einer Senderate von $0,1\ 1/ms$ der verfügbaren Zeit zwischen zwei initialen Sendeereignissen und bewirkt somit eine Vollauslastung des Senders. Für die empfängerinitiierten Protokolle liegt diese Schranke mit 700 für K-NACK respektive mit 3000 Gruppenmitgliedern für K-NAPP höher. Dies kann durch eine überschlagende Betrachtung ebenfalls verifiziert werden, wenn man annimmt, dass bei 10% Nachrichtenverlustwahrscheinlichkeit im ersten Durchgang für K-NACK durchschnittlich 70 negative Quittungen zu verarbeiten sind. In den folgenden Durchgängen sind es entsprechend weniger. Den numerischen

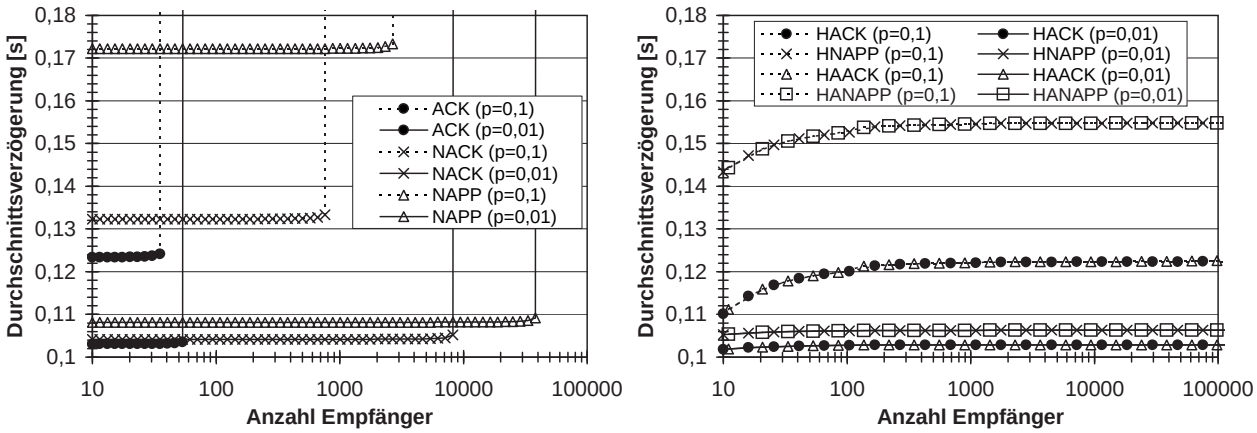



*Abbildung 4.19: Durchschnittsverzögerung flacher und hierarchischer Protokolle*

Ergebnissen von K-NAPP liegt die Annahme zugrunde, dass die Wahrscheinlichkeit für simultanes Senden von negativen Quittungen bei 20% liegt (siehe Abschnitt 4.6.2), d.h. es werden fünfmal weniger Quittungen gesendet. Folglich steigt die unterstützte Gruppengröße etwa um diese Größe. Eine geringere Nachrichtenverlustwahrscheinlichkeit führt zu weniger Übertragungswiederholungen und bei den empfängerinitiierten Protokollen auch zu weniger Quittungen, so dass von allen Protokollen eine größere Teilnehmerzahl unterstützt werden kann.

Werden die Protokolle innerhalb ihres Skalierbarkeitsbereichs, d.h. in dem Bereich zwischen einem Teilnehmer und der Teilnehmerzahl an der die Verzögerungen stark ansteigenden, näher betrachtet fällt auf, dass die Klasse K-ACK die geringsten Verzögerungen aufweist. Dies ist ein wesentlicher Grund dafür, die Klasse K-ACK und allgemein senderinitiierte Protokolle aufgrund ihrer begrenzten Skalierbarkeit nicht vorschnell außen vor zu lassen. Die Erklärung dieses Verhaltens erfolgt später in Zusammenhang mit Abbildung 4.21.

Hierarchische Protokolle bieten bei kleiner Lokalgruppenüberlappung unbegrenzte Skalierbarkeit für beliebige Gruppengrößen. Diese Abbildung und alle Folgenden zeigen die Ergebnisse für eine Lokalgruppenüberlappung von 0,001. Zwischen 10 und 1000 Empfängern ist ein Anstieg der Verzögerungen zu beobachten. In diesem Bereich gewinnt der Kontrollbaum schnell an Höhe, was sich in gestiegenen Verzögerungszeiten bei eventuell notwendigen Übertragungswiederholungen über mehrere Stufen im Kontrollbaum bemerkbar macht. Der Höhenzuwachs nimmt mit steigender Gruppengröße lediglich logarithmisch zu, so dass eine Abflachung der Zeitzuwächse beobachtet werden kann. Beim direkten Vergleich zwischen flachen und hierarchischen Protokollen wird ersichtlich, dass die hierarchischen Protokolle auch innerhalb des Skalierbarkeitsbereichs flacher Protokolle geringere Nachrichtenverzögerungen erreichen.[5] Dies ist hauptsächlich durch die geringeren Nachrichtenverzögerungen

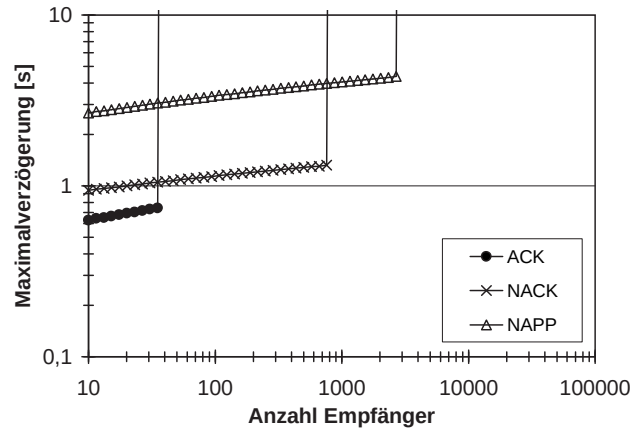
[5]Mit Ausnahme von Klasse K-NACK, für die keine äquivalente hierarchische Klasse untersucht wurde.

für Übertragungswiederholungen innerhalb einer lokalen Gruppe begründet. Nicht nur die Nachrichtenlaufzeiten sind dadurch geringer, sondern auch die Zeitgeber können kürzer eingestellt werden, was den Verzögerungszeiten ebenfalls zugute kommt.

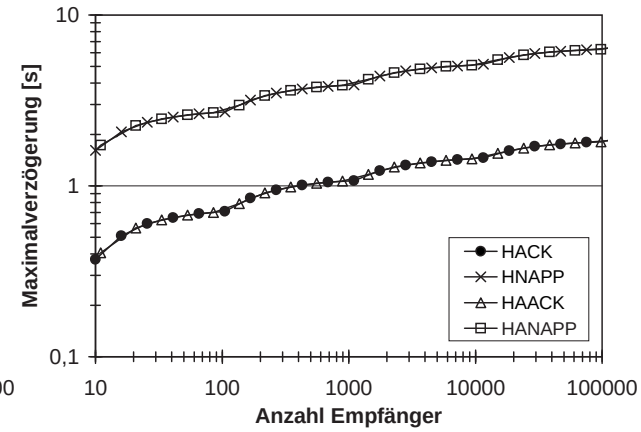
Im Vergleich mit der Bandbreiten– und CPU-Analyse sind die Verzögerungszeiten innerhalb des Skalierbarkeitsbereichs eines Protokolls kaum von der Lokalgruppenüberlappung abhängig. Wird diese allerdings zu hoch, so werden die Knoten überlastet, was wie in den flachen Protokollklassen zu steil ansteigenden Verzögerungszeiten führt. Beispielsweise besitzt Protokollklasse K-HACK mit einer Lokalgruppenüberlappung $p_l$ von 0,01 und einer Nachrichtenverlustwahrscheinlichkeit von 1% bis 70000 Empfängern die annähernd gleiche Verzögerung wie ohne Lokalgruppenüberlappung oder wie in Abbildung 4.19 mit $p_l = 0,001$ dargestellt ist. Mehr als 70000 Empfänger können hingegeben mit $p_l = 0,01$ nicht unterstützt werden, da sich ein steiler Anstieg der Verzögerung ergibt.

Abbildung 4.20 veranschaulicht für alle untersuchten Protokollklassen die Maximal– und Umlaufverzögerungen mit $\gamma = 0.999$. Zu beachten ist, dass beide Achsen logarithmisch skaliert sind. Die quantitativen Resultate unterscheiden sich beträchtlich von den Durchschnittsverzögerungen. Während die Durchschnittsverzögerungen — vor allem bei einer geringen Nachrichtenverlustwahrscheinlichkeit — nur knapp oberhalb der Nachrichtenverzögerung im Netzwerk angesiedelt sind ($\tau = 100ms$), sind die Maximal– und Umlaufverzögerungen deutlich höher. Zudem ist eine erhöhte Abhängigkeit von der Gruppengröße zu beobachten, da alle Protokolle mit Ausnahme der Umlaufverzögerung von K-HACK und K-HNAPP einen Anstieg mit zunehmender Teilnehmerzahl verzeichnen. Für die Maximalverzögerung ist dies durch die Notwendigkeit begründet, bei allen Empfängern die Nachricht erfolgreich zu empfangen, was mit zunehmender Teilnehmerzahl mehr Übertragungswiederholungen erfordert. Für die Umlaufverzögerung gilt dies analog. Zudem ist für K-HAACK und K-HANAPP die schrittweise Weiterleitung der aggregierten Quittungen durch die TRS-Hierarchie zu bedenken. Maßgebend ist vor allem die Anzahl der Hierarchiestufen im Kontrollbaum, mit denen die Umlaufverzögerung stetig zunimmt. Für K-HACK und K-HNAPP wurde die Umlaufverzögerung lediglich innerhalb einer lokalen Gruppe bestimmt, bedingt durch die fehlenden aggregierten Quittungen. Folglich ist keine Zunahme mit der Anzahl Hierarchiestufen gegeben. Im Gegenteil kann sogar eine leichte Abnahme der Umlaufverzögerung mit wachsender Gruppengröße beobachtet werden, da durch die größere Anzahl lokaler Gruppen die Durchschnittsverzögerung innerhalb einer lokalen Gruppe abnimmt.

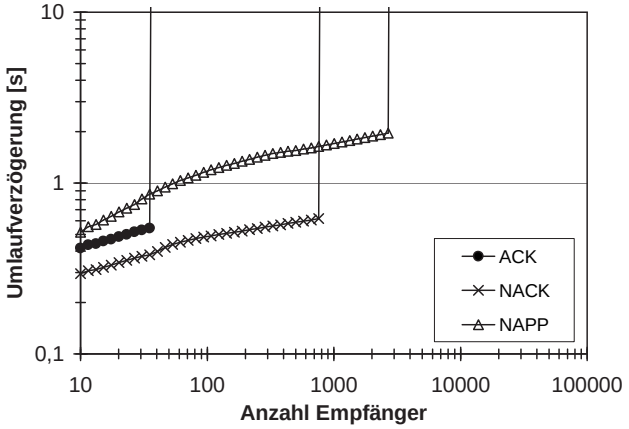
Nun wird näher erörtert, warum sowohl bei flachen als auch bei hierarchischen Protokollen die senderinitiierten Vertreter eine geringere Verzögerung aufweisen. Dazu zeigt Abbildung 4.21 die durchschnittliche Verzögerung und die Umlaufverzögerung in Abhängigkeit von der Senderate bei 10000 Empfängern. Die Maximalverzögerung ist nicht abgebildet, da die Ergebnisse ähnlich der Umlaufverzögerung ausfallen. Es zeigt sich eine proportionale Abhängigkeit der Verzögerungen empfängerinitiierter Protokolle von der Senderate. Wodurch kommt diese Abhängigkeit zustande? Die Erklärung



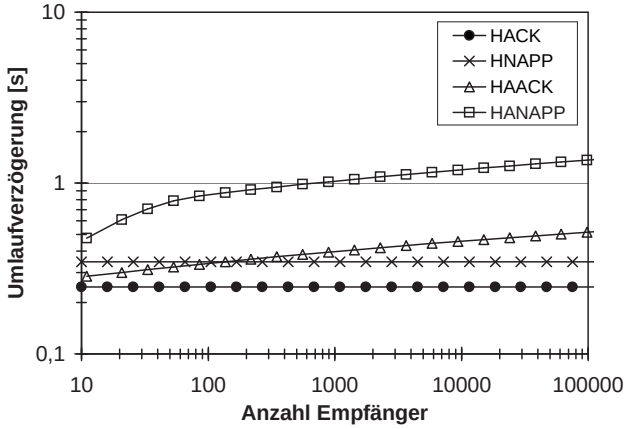

(a) Maximalverzögerung flacher Protokolle




(b) Maximalverzögerung hierarchischer Protokolle




(c) Umlaufverzögerung flacher Protokolle



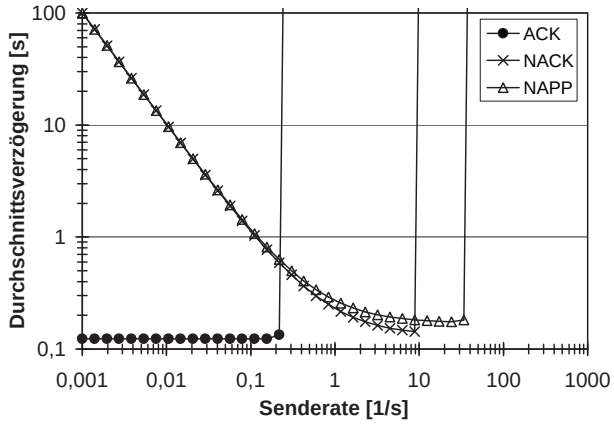

(d) Umlaufverzögerung hierarchischer Protokolle

*Abbildung 4.20: Maximalverzögerung und Umlaufverzögerung bei einer Nachrichtenverlustwahrscheinlichkeit von 10%*

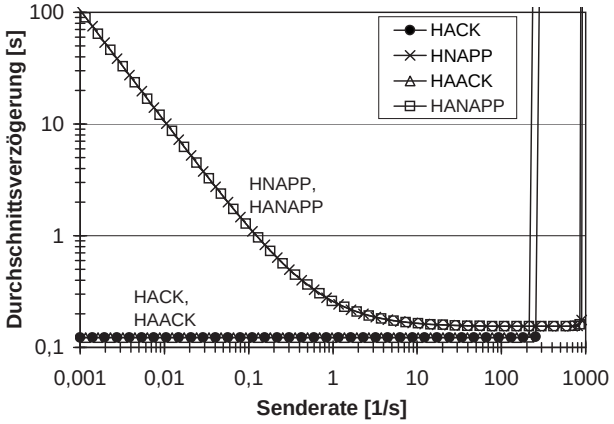
liefert die empfängerbasierte Fehlerdetektion. Ein Empfänger kann eine fehlende Nachricht erst mit dem fehlerfreien Empfang einer nachfolgenden Nachricht und dem damit bemerkten Sprung in der Sequenznummer detektieren. Ist die Senderate ausgesprochen niedrig, dann dauert die Fehlererkennung entsprechend lange, was schließlich die Verzögerungen anwachsen lässt. Für niedrige Senderaten ist diese der dominierende Faktor bei den Verzögerungen.[6]

Eine einfache Überlegung kommt bei niedrigen Senderaten den tatsächlichen Ergebnissen der Analyse recht nahe. Bei einer Senderate von $0,1\ 1/s$ liegen zwischen dem Empfang von zwei Paketen 10*s*, was der Zeit einen einzelnen Paketverlust zu erkennen entspricht. Ein Vergleich mit Abbildung 4.21c

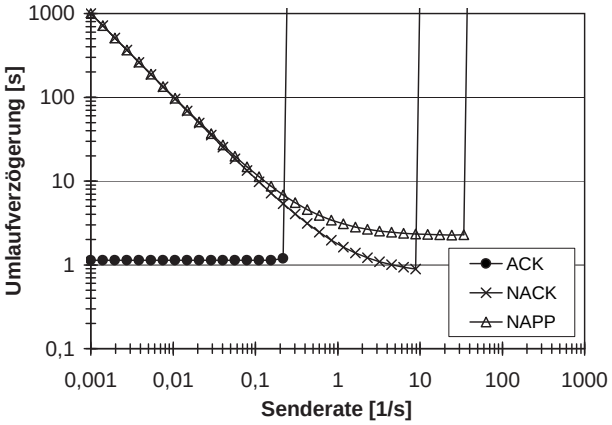
[6]Empfängerinitiierte Protokolle können diesen Nachteil umgehen, indem ein Sender bei ausbleibenden Nutzdaten leere Nachrichten versendet, um eine Mindestnachrichtenrate aufrecht zu erhalten (siehe Beschreibung von MTP in Abschnitt 2.5.3.2). Allerdings wird dadurch der Bandbreitenbedarf erhöht.



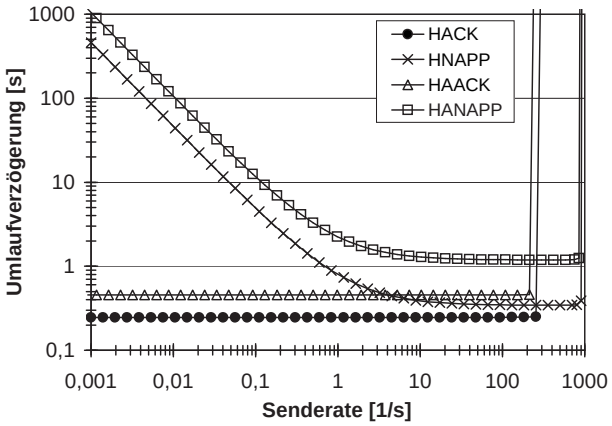

(a) Durchschnittsverzögerung flacher Protokolle




(b) Durchschnittsverzögerung hierarch. Protokolle




(c) Umlaufverzögerung flacher Protokolle

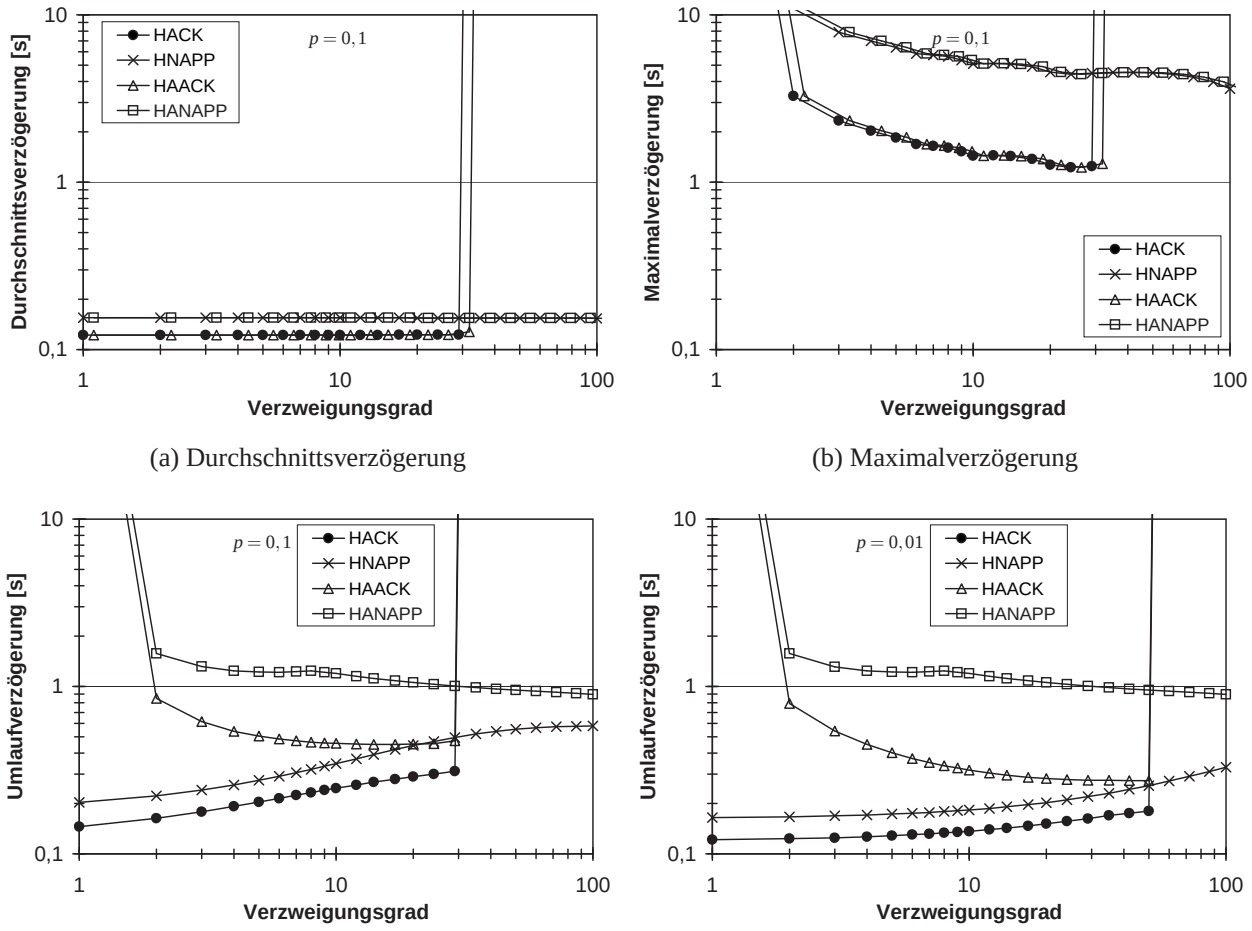



(d) Umlaufverzögerung hierarch. Protokolle

*Abbildung 4.21: Abhängigkeit der Verzögerung von der Senderate bei einer Nachrichtenverlustwahrscheinlichkeit von 10%*

bestätigt, dass in diesem Fall die Umlaufverzögerung ungefähr 10*s* beträgt (die Maximalverzögerung liegt ebenfalls bei ungefähr 10*s*). Diese ist lediglich zu einem kleinen Teil von der Nachrichtenverlustwahrscheinlichkeit bestimmt. Soll das Ergebnis der Durchschnittsverzögerung betrachtet werden, muss berücksichtigt werden, dass bei einer Nachrichtenverlustwahrscheinlichkeit von 10% nur 10% der Pakete eine Übertragungswiederholung benötigen. Die restlichen 90% der Pakete werden unmittelbar von der initialen Übertragung empfangen, ohne dass eine Fehlerdetektion die Verzögerungen erhöht. Deshalb sind die Ergebnisse um ungefähr den Faktor 10 niedriger, was in Abbildung 4.21a ersichtlich ist. Die Fehlerdetektion der senderinitiierten Protokolle ist unabhängig von der Senderate, da direkt auf ausbleibende Quittungen reagiert werden kann und deshalb meistens wesentlich schneller.

Desweiteren wird aus der Abbildung ersichtlich, dass flache Protokolle nicht nur bezüglich der maximalen Gruppengröße eine Schranke aufweisen, sondern auch bezüglich der maximalen Senderate.




(a) Durchschnittsverzögerung

(b) Maximalverzögerung

(c) Umlaufverzögerung mit $p = 0,1$

(d) Umlaufverzögerung mit $p = 0,01$

*Abbildung 4.22: Abhängigkeit der Verzögerung vom Verzweigungsgrad*

Ebenso besitzen die hierarchischen Protokolle eine Schranke für die maximale Senderate. Allerdings ist diese wesentlich höher und zumindest für die empfängerinitiierten Protokolle bereits sehr nahe an der theoretischen Obergrenze von zwei Nachrichten pro Millisekunde in diesem Szenario. Die Obergrenze ergibt sich aus dem angenommenen Zeitbedarf von $0,5ms$ für das Senden einer Datennachricht, was maximal zwei Nachrichten pro Millisekunde erlaubt.

Die Abhängigkeit der Verzögerungen vom Verzweigungsgrad wird in Abbildung 4.22 dargestellt. Die Ergebnisse basieren auf der Annahme einer Gruppengröße von 10000 Empfängern. Die Nachrichtenverlustwahrscheinlichkeit beträgt 10% und die Senderate $0,1$ $1/ms$. Abbildung 4.22a zeigt dazu die Durchschnittsverzögerung, die keine interessanten Abhängigkeiten zeigt. Einzig fällt auf, dass die senderinitiierten Protokolle K-HACK und K-HAACK ab ungefähr 30 Gruppenmitgliedern pro lokaler Gruppe in diesem Szenario überlastet sind. Abbildung 4.22b zeigt die Maximalverzögerung, die dagegen interessantere Ergebnisse liefert. Demnach verringert sich diese mit steigendem

Verzweigungsgrad. Bereits zur Abbildung 4.20 wurde erläutert, dass die Maximalverzögerung von der Anzahl an Hierarchiestufen im Kontrollbaum abhängt. Im ungünstigsten Fall müssen sequentiell über alle Hierarchiestufen hinweg, beginnend beim Sender, Übertragungswiederholungen durchgeführt werden. Daraus resultiert die starke Abhängigkeit vom Verzweigungsgrad — zwischen den Verzweigungsgraden 2 und 30 ergibt sich immerhin eine Verringerung der Verzögerung um bis zu 70% — denn ein großer Verzweigungsgrad ergibt niedrige Kontrollbäume mit wenigen Hierarchiestufen. Die Umlaufverzögerung ist sowohl mit 10% als auch mit 1% Nachrichtenverlustwahrscheinlichkeit abgebildet. Für die Protokolle mit aggregierten Quittungen zeigt die Umlaufverzögerung den gleichen Trend wie die Maximalverzögerung. Protokolle ohne aggregierte Quittungen profitieren nicht durch weniger Hierarchiestufen, sondern zeigen im Gegenteil, durch das steigende Nachrichtenaufkommen bei größeren lokalen Gruppen, eine wachsende Umlaufverzögerung.

In Abschnitt 4.8 werden die interessantesten Ergebnisse zur Leistungsanalyse der Protokollklassen und insbesondere zum Verzweigungsgrad und den Konsequenzen für den TRS-Dienst zusammengefasst.

## 4.7 Simulationen

In den folgenden Abschnitten werden Simulationen zuverlässiger Transportprotokolle vorgestellt. Die Simulationen haben das Ziel, die numerischen Ergebnisse der Analyse zu bestätigen. Für den Aufbau der Kontrollbäume wird dabei der TRS-M-Dienst eingesetzt, mit der Kontrollbaumhöhe als zu optimierende Metrik bei der Gruppenführerauswahl, da nur dieser den vorgegebenen maximalen Verzweigungsgrad auch tatsächlich erreicht. Die schwankenden Verzweigungsgrade des entstehenden Kontrollbaums vor allem von ERS und ERA würden einen Vergleich zumindest erschweren oder gar unmöglich machen, bedenkt man dessen großen Einfluss auf die Resultate. Ansonsten gelten die Ausführungen zum Simulationssystem aus Abschnitt 3.10 hier ebenfalls.

### 4.7.1 Testumgebung

Zum Vergleich der numerischen Ergebnisse der Analyse mit Simulationsergebnissen wurden repräsentativ die Protokolle SRM [Floyd et al. 1997], RMTP [Paul et al. 1997] und TMTP [Yavatkar et al. 1995] implementiert, von denen RMTP und TMTP zur Klasse der hierarchischen Transportprotokolle gehören. Während RMTP positive Quittungen verwendet, basieren SRM und TMTP auf negativen Quittungen mit Quittungsunterdrückung. Alle drei Protokolle wurden in Abschnitt 2.5.3 beschrieben. In allen Protokollen mussten allerdings, abweichend von deren Spezifikationen, Änderungen vorgenommen werden. Diese Änderungen betrafen sowohl die einfachere Implementierbarkeit als auch Einschränkungen, um Messungen überhaupt sinnvoll durchführen zu können.

Die wesentlichste Änderung des RMTP-Protokolls betrifft den Mechanismus für Übertragungswiederholungen. RMTP spezifiziert dazu einen „Subcast“, der durch Router-Unterstützung ermöglicht werden muss. Im Gegensatz dazu verwendet die Simulationsimplementierung einen normalen Multicast an die gesamte Gruppe, dessen Ausbreitungsbereich begrenzt wird. RMTP definiert keinen Mechanismus, um Gruppenführer zu bestimmen. Die Annahme ist, dass Gruppenführer manuell an strategisch günstigen Positionen im Netzwerk aufgestellt werden. Im Gegensatz dazu bietet bei der hier verwendeten Implementierung prinzipiell jeder Teilnehmer der Multicast-Gruppe an, die Gruppenführerrolle zu übernehmen. Die letzte Änderung betrifft das Senden von Quittungen. RMTP sieht vor, dass diese zum Zweck der besseren Skalierbarkeit periodisch gesendet werden, anstatt nach jedem Datenempfang. Die Simulationsimplementierung sendet Quittungen dagegen nach jedem Datenempfang. Dies verbessert die Auslieferungszeit, da Übertragungswiederholungen früher durchgeführt werden, erhöht allerdings das Verkehrsaufkommen. Durch die beschriebenen Anpassungen wird eine Vergleichbarkeit mit der Analyse ermöglicht.

In allen drei Protokollen wurde eine ratenbasierte Flusskontrolle implementiert. Abweichend von der Spezifikation werden bei TMTP und RMTP aggregierte Quittungen anstatt einfacher positiver Quittungen verwendet. Positive Quittungen werden gesendet, sobald das entsprechende Datenpaket empfangen wurde, während aggregierte Quittungen erst gesendet werden, wenn auch alle Kindknoten den Empfang bestätigt haben. Dies besitzt den Vorteil, dass auf einen Knotenausfall reagiert werden kann wie in Abschnitt 4.2 ausgeführt wurde. In TMTP als empfängerinitiiertem Protokoll ist in der Spezifikation vorgesehen, dass die zusätzlichen positiven Quittungen periodisch gesendet werden. Deren Aufgabe, sowohl deterministische Zuverlässigkeit garantieren zu können als auch fensterbasierte Flusskontrolle, wird in der Simulationsimplementierung bereits durch die aggregierten Quittungen erfüllt. Die aggregierten Quittungen für TMTP werden nach jeder Datenübertragung gesendet, um die gemessenen Verzögerungen mit der Analyse vergleichen zu können. Die Konsequenz aus der Verwendung von aggregierten Quittungen ist, dass die Simulationsimplementierung von RMTP zur Klasse K-HAACK gehört und TMTP zur Klasse K-HANAPP. SRM gehört zur Klasse K-NAPP. Damit wurde mit dem am besten skalierbaren flachen Protokoll und zwei hierarchischen Protokollen eine repräsentative Auswahl getroffen, um die Analyseergebnisse zu bestätigen.

Alle Simulationen basieren auf Netzwerken bestehend aus 250 oder 1000 Knoten, die mit Tiers erzeugt wurden. Die Bandbreite beträgt durchgängig 100 MBit/s für alle Verbindungen. Die Verzögerungen betragen zufällig pro Verbindung zwischen 1 und 42 Millisekunden.[7] Alle Knoten im Netzwerk verwenden DVMRP-Routing [Waitzman et al. 1988]. Um Nachrichtenverlust zu simulieren gibt es prinzipiell zwei Möglichkeiten. Entweder wird für jede Verbindung auf Sicherungsschicht, d.h. für zwei direkt benachbarte Knoten, eine Paketverlustwahrscheinlichkeit definiert oder die Nachrich-

[7] Für Verzögerungen im Internet siehe [Carter & Crovella 1996] und [Theilmann 2000] und die Ausführungen in Abschnitt 3.10.

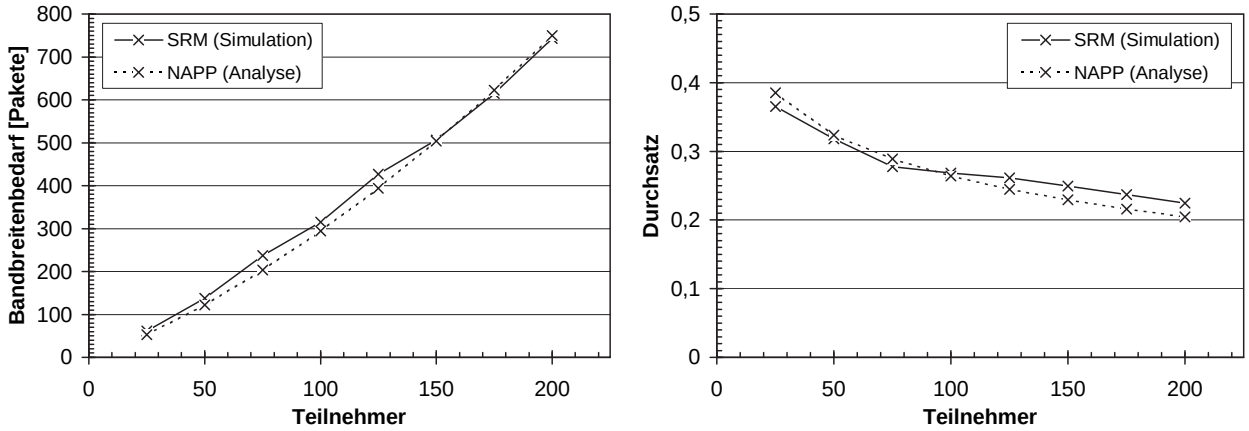



*Abbildung 4.23: Vergleich des Bandbreitenbedarfs zwischen der Simulation von SRM und der Analyse von K-NAPP a) Gesamtbandbreitenbedarf und b) bandbreitenbeschränkter Durchsatz*

tenverlustwahrscheinlichkeit wird direkt zwischen den Ende-zu-Ende Verbindungen auf Transportschicht definiert. Obwohl der zweite Ansatz einfacher zu implementieren ist und die Vorgabe einer definierten und für alle Simulationen und Netzwerke festen Ende-zu-Ende Nachrichtenverlustwahrscheinlichkeit erlaubt, wurde aus gutem Grund der erste Ansatz implementiert. Auch in realen Netzwerken tritt der Nachrichtenverlust bei Verbindungen auf Sicherungsschicht und nicht auf höheren Schichten auf. Dies führt bei Multicast-Nachrichten zu räumlich abhängigen Verlusten wie sie auch in der Realität auftreten (siehe Abschnitt 4.4), wenn eine Nachricht verloren geht, die im Multicast-Spannbaum noch ein oder mehrere Duplikationen durchlaufen hätte. In den folgenden Simulationen wurde eine Paketverlustwahrscheinlichkeit von 0,02 pro Verbindung auf Sicherungsschicht konfiguriert. Dies resultierte in einer gemessenen durchschnittlichen Ende-zu-Ende Nachrichtenverlustwahrscheinlichkeit von 12%. Kontrollmessungen mit Paketverlustwahrscheinlichkeit 0,002 pro Verbindungen resultierten in ungefähr 1% Nachrichtenverlustwahrscheinlichkeit und ergaben qualitativ die selben Resultate. Ein Nachteil dieser Vorgehensweise soll nicht unerwähnt bleiben. Die Ende-zu-Ende Nachrichtenverlustwahrscheinlichkeit hatte nur im Mittel die angegebenen Werte, schwankte aber in einzelnen Simulationen um bis zu 40%. Dadurch weisen auch die Ergebnisse Schwankungen auf.

## 4.7.2 Simulationsresultate

### 4.7.2.1 SRM

In Abbildung 4.23 sind die Ergebnisse der Gesamtbandbreite und des bandbreitenbeschränkten Durchsatzes zwischen Simulation und Analyse vergleichend dargestellt. Teilabbildung a) zeigt sowohl bei der Analyse als auch bei der Simulation einen leicht überproportionalen Anstieg des Gesamtbandbreitenbedarfs mit steigender Teilnehmerzahl. Daraus lässt sich ableiten, dass jeder Teilnehmer mit

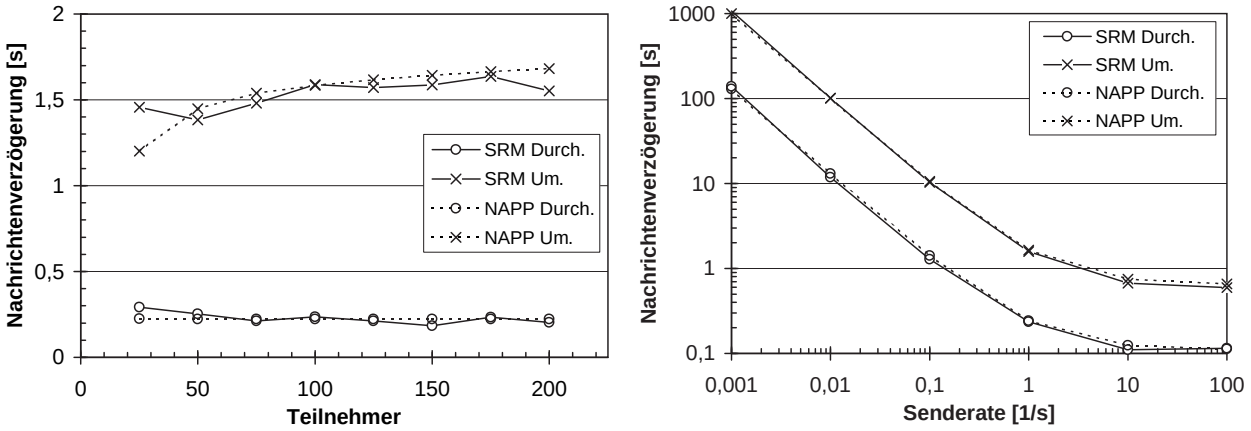



*Abbildung 4.24: Vergleich der Durchschnitts– und Umlaufverzögerung zwischen der Simulation von SRM und der Analyse von K-NAPP a) in Abhängigkeit von der Teilnehmerzahl und b) in Abhängigkeit von der Senderate*

steigender Teilnehmerzahl etwas mehr Nachrichten zu verarbeiten hat. Teilabbildung b) mit der Darstellung des normierten Durchsatzes bestätigt dies wiederum für die Analyse als auch die Simulation mit abnehmendem maximalen Durchsatz bei steigender Teilnehmerzahl, was die eingeschränkte Skalierbarkeit flacher Protokolle bestätigt. Nicht nur die Tendenz ist durch die Simulation untermauert, sondern auch die absoluten Werte zeigen lediglich geringe Abweichungen.

Zur Validierung der Verzögerungsanalyse werden sowohl die Durchschnitts– als auch die Umlaufverzögerung dargestellt. Die durchschnittliche Verzögerung auf Transportschicht ($\tau$) des Simulationsszenarios wurde mit 74 Millisekunden gemessen. Die Senderate ($\lambda$) beträgt bei Simulationen, in denen die Teilnehmerzahl variiert, eine Nachricht pro Sekunde. Bei variierender Senderate beträgt die Teilnehmerzahl 100.

Abbildung 4.24a zeigt die Verzögerungen in Abhängigkeit der Teilnehmerzahl. Für die Durchschnittsverzögerung ergibt sich praktisch keine Abhängigkeit von der Teilnehmerzahl und damit eine hohe Übereinstimmung zwischen Simulation und Analyse. Für die Umlaufverzögerung ist eine merkliche quantitative Abweichung feststellbar. Allerdings kann noch eine leicht steigende Tendenz mit steigender Teilnehmerzahl beobachtet werden, was die Analyseergebnisse bestätigt. In Teilabbildung b) wird die Abhängigkeit von der Senderate untersucht. Hier ergibt sich ebenfalls die aus der Analyse vorhergesagte Abhängigkeit empfängerinitiierten Protokolle von der Senderate. Beide gemessene Verzögerungszeiten weisen zudem nur geringe quantitative Abweichungen zur Analyse auf.

Auf eine Darstellung der Maximalverzögerung wurde verzichtet. Die Maximalverzögerung wies höhere Schwankungen auf, da diese durch den langsamsten bzw. fehlerträchtigsten Empfänger, also eines einzelnen Empfängers, bestimmt wird. Bedingt durch diese Schwankungen ist ein Vergleich mit der Analyse lediglich mit einer großen Anzahl von Simulationsdurchläufen und anschließender

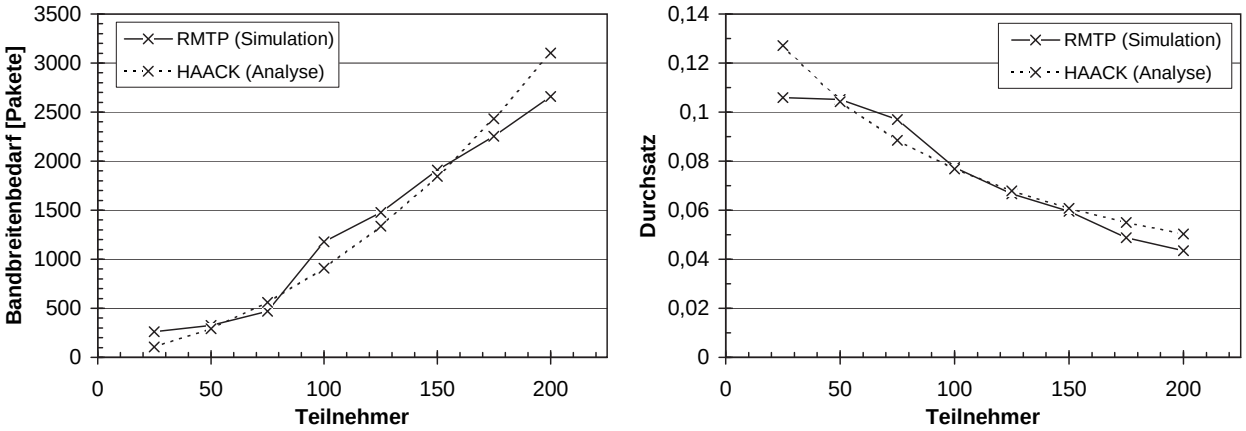



*Abbildung 4.25: Vergleich des Bandbreitenbedarfs zwischen der Simulation von RMTP und der Analyse von K-HAACK a) Gesamtbandbreitenbedarf und b) bandbreitenbeschränkter Durchsatz*

Mittelung der Ergebnisse sinnvoll, was aber durch die langen Simulationszeiten nicht möglich war. Es sei allerdings darauf hingewiesen, dass trotz schwankender Messwerte der Maximalverzögerung in Einzelsimulationen, das vorausgesagte qualitative Verhalten aus der Analyse von den Simulationen auch in diesem Fall bestätigt wurde.

#### 4.7.2.2 RMTP

In Abbildung 4.25 sind die Resultate des Bandbreitenbedarfs veranschaulicht. Alle Ergebnisse wurden mit einem maximalen Verzweigungsgrad des Kontrollbaums von zehn ermittelt. Für die Lokalgruppenüberlappung wurde mit durchschnittlich 30% ein vergleichsweise großer Wert gemessen. Dies ist durch die ungewöhnlich dichten Gruppen verursacht (40% der Netzwerkknoten gehören der Multicast-Gruppe an), was eine sinnvolle Etablierung lokaler Gruppen und Begrenzung des Auslieferungsbereichs lokaler Übertragungswiederholungen mittels des IP-TTLs erschwert.

Sowohl der maximale Durchsatz als auch der Gesamtbandbreitenbedarf in Abbildung 4.25 zeigen bis auf zufällige Schwankungen eine hohe Korrelation zwischen Analyse und Simulation. Verursacht durch die vergleichsweise hohe Lokalgruppenüberlappung sinkt auch hier der Durchsatz mit zunehmender Teilnehmerzahl. Die Simulationsergebnisse der Verzögerungen sind in Abbildung 4.26 dargestellt und unterstreichen ebenfalls die Analyseergebnisse. Die durchschnittliche Verzögerung weist, sowohl für unterschiedliche Teilnehmerzahlen als auch Senderaten, im Gegensatz zu empfängerinitiierten Protokollen, ein konstantes Verhalten auf. Die quantitativen Resultate zur Umlaufverzögerung zeigen teilweise kleine Abweichungen von den Analysen, dennoch kann der Trend zu zunehmender Umlaufverzögerung mit steigender Teilnehmerzahl bestätigt werden.

Abbildung 4.27 zeigt schließlich die Abhängigkeit des Bandbreitenbedarfs und der Verzögerungen vom maximalen Verzweigungsgrad des Kontrollbaums. Die Multicast-Gruppe besteht aus 200 Teil-

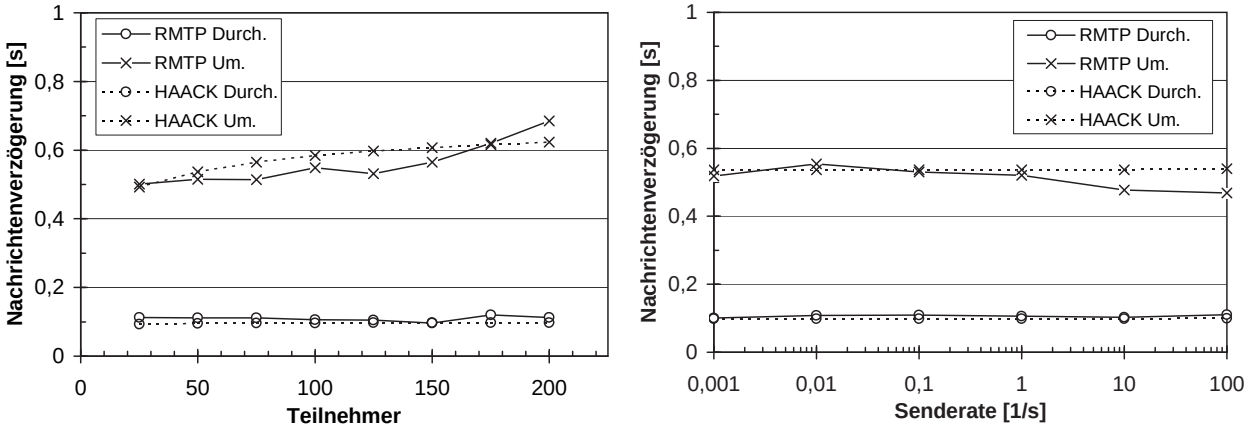



*Abbildung 4.26: Vergleich der Durchschnitts– und Umlaufverzögerung zwischen der Simulation von RMTP und der Analyse von K-HAACK a) in Abhängigkeit der von Teilnehmerzahl und b) in Abhängigkeit von der Senderate*

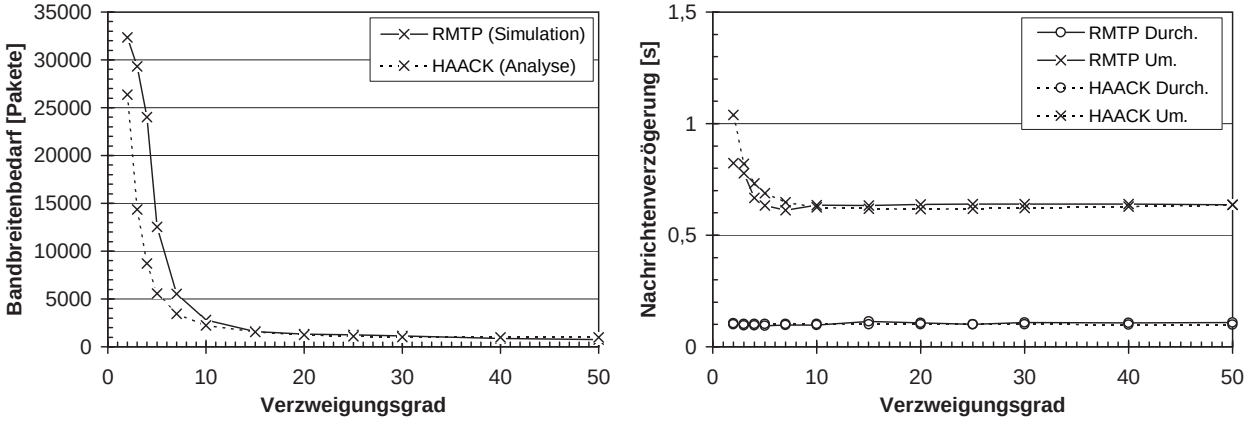



*Abbildung 4.27: Vergleich zwischen der Simulation von RMTP und der Analyse von K-HAACK a) Gesamtbandbreitenbedarf und b) Verzögerungen in Abhängigkeit des Verzweigungsgrads*

nehmern und die Senderate beträgt eine Nachricht pro Sekunde. Der gemessene Bandbreitenbedarf bestätigt die aus der Analyse prognostizierte Abnahme mit zunehmendem Verzweigungsgrad bei hoher Lokalgruppenüberlappung. Die Simulation zeigt diesen Effekt deutlich ausgeprägter als die Analyse, was auf nicht konstante Lokalgruppenüberlappungen und Nachrichtenverlustwahrscheinlichkeiten zurückzuführen ist. Auch die gemessene Durchschnitts– und Umlaufverzögerung, die in Abbildung 4.27b illustriert ist, bestätigt die Analyseergebnisse. Die Durchschnittsverzögerung wird durch den Verzweigungsgrad praktisch nicht beeinflusst, da die durchaus beeinflusste Verzögerung für Übertragungswiederholungen auf die Durchschnittsverzögerung nur sehr begrenzten Einfluss hat. Ganz anders ist dies bei der Umlaufverzögerung. Hier zeigt auch die Simulation einen deutlichen Rückgang der Verzögerung, vor allem beim Erhöhen des maximalen Verzweigungsgrads im einstel-

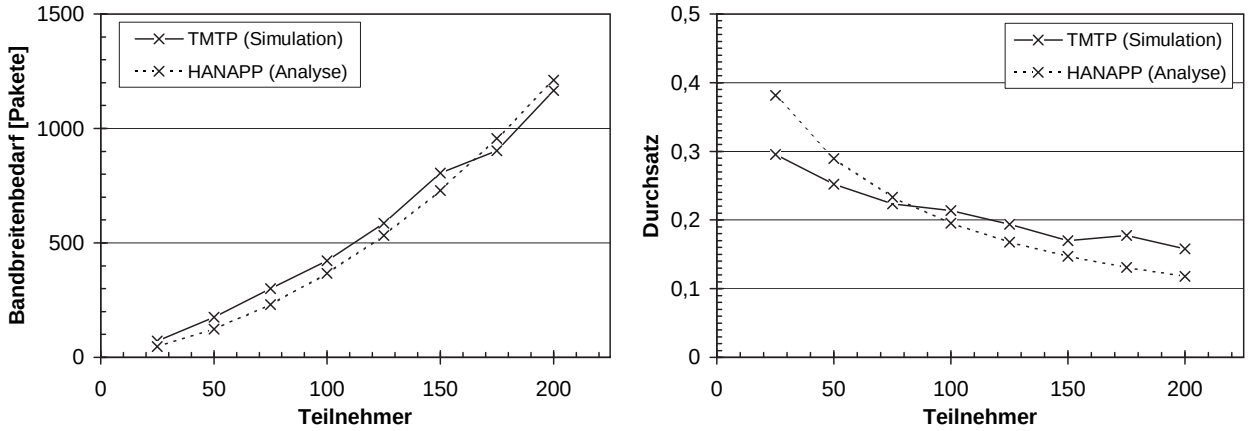



*Abbildung 4.28: Vergleich des Bandbreitenbedarfs zwischen der Simulation von TMTP und der Analyse von K-HANAPP a) Gesamtbandbreitenbedarf und b) bandbreitenbeschränkter Durchsatz*

ligen Bereich. Bei einem größeren Verzweigungsgrad nimmt die Kontrollbaumhöhe bei dieser geringen Gruppengröße nur noch wenig ab, so dass kein weiterer signifikanter Einfluss festgestellt werden kann.

#### 4.7.2.3 TMTP

Schließlich werden nachfolgend die Ergebnisse von TMTP mit der Analyse verglichen. In Abbildung 4.28 sind die Ergebnisse der Bandbreite aufgetragen. Der Gesamtbandbreitenbedarf in Teilabbildung a) zeigt große Übereinstimmung zwischen Simulation und Analyse. In Teilabbildung b) ist eine Abweichung in der Steigung beider Kurven auffällig, was durch eine sich ändernde Lokalgruppenüberlappung verursacht wird. Diese nimmt für die Simulation mit steigender Teilnehmerzahl ab, da geeignetere lokale Gruppen gebildet werden können, während für die Analyse der Durchschnittswert angenommen wurde. Die Umlaufverzögerung in Abbildung 4.29 zeigt bei der Analyse mit steigenden Teilnehmerzahlen eine zunehmende Tendenz. In der Simulation lässt sich diese Tendenz, allerdings nur sehr schwach, ebenfalls ermitteln. Wiederum ist die Durchschnittsverzögerung unabhängig von den Teilnehmerzahlen. In Teilabbildung b) zeigen sich übereinstimmend für Analyse und Simulation die fallenden Verzögerungszeiten mit steigender Senderate, bedingt durch die raschere Fehlererkennung. Interessant sind abschließend die Ergebnisse für einen variablen Verzweigungsgrad des Kontrollbaums. Das Ergebnis der Gesamtbandbreite unterstreicht die fallende Tendenz bei steigendem Verzweigungsgrad und damit die Analyseergebnisse. Auch die Durchschnitts- und Umlaufverzögerung bestätigen die Analyseergebnisse.

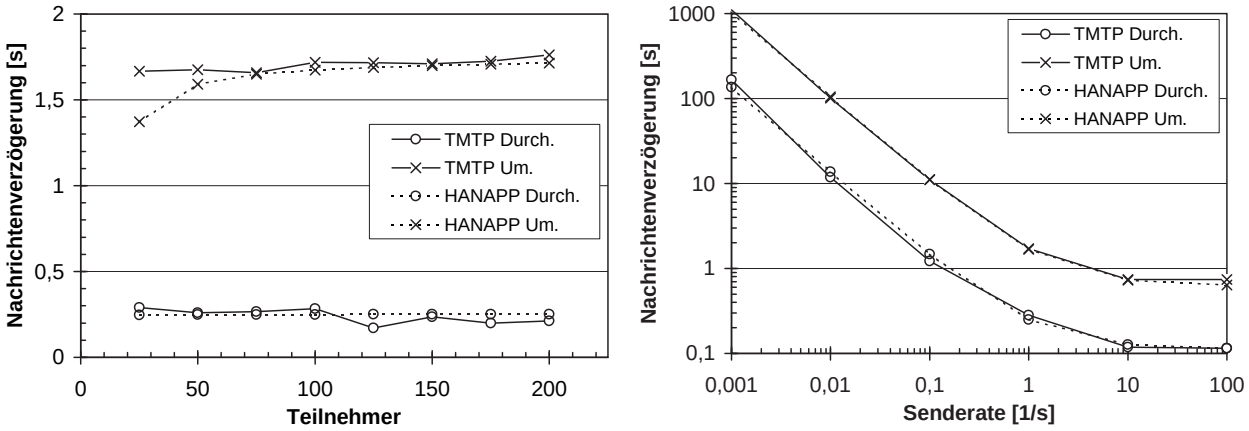



*Abbildung 4.29: Vergleich der Durchschnitts– und Umlaufverzögerung zwischen der Simulation von TMTP und der Analyse von K-HANAPP a) in Abhängigkeit von der Teilnehmerzahl und b) in Abhängigkeit von der Senderate*

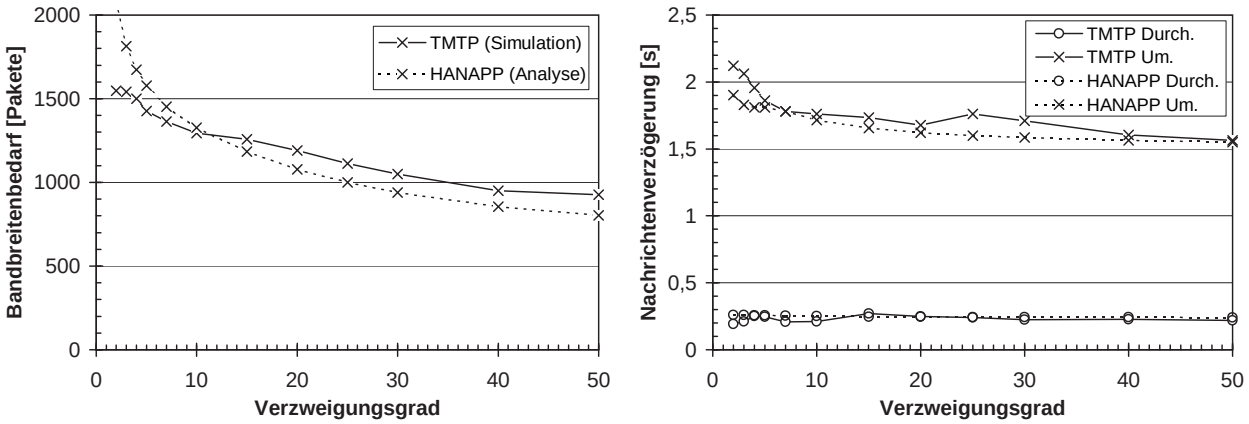



*Abbildung 4.30: Vergleich zwischen der Simulation von TMTP und der Analyse von K-HANAPP a) Gesamtbandbreitenbedarf und b) Verzögerungen in Abhängigkeit des Verzweigungsgrads*

#### 4.7.2.4 Vergleich zwischen einem flachen und einem hierarchischen Protokoll

Abschließend soll das wichtigste Resultat aus der Analyse, die bessere Skalierbarkeit von hierarchischen Protokollen, überprüft werden. Dazu werden in Abbildung 4.31 der Bandbreitenbedarf und der bandbreitenbeschränkte Durchsatz der beiden skalierbarsten Vertreter ihrer Klasse, SRM und TMTP, verglichen. Um den Einfluss der Lokalgruppenüberlappung auszuschließen, wurde jeder lokalen Gruppe eine eigene Multicast-Adresse zugewiesen. Der Bandbreitenbedarf von TMTP wächst signifikant langsamer mit steigender Teilnehmerzahl als der von SRM. Bereits kleinste Gruppen profitieren von den lokalen Übertragungswiederholungen. Der Durchsatz hingegen resultiert für kleinste Gruppen noch in einem Vorteil von SRM gegenüber TMTP, was durchaus konsistent mit den Analyseergebnissen ist. Der Grund liegt in der Doppelbelastung der Gruppenführer, die zugleich Sender und

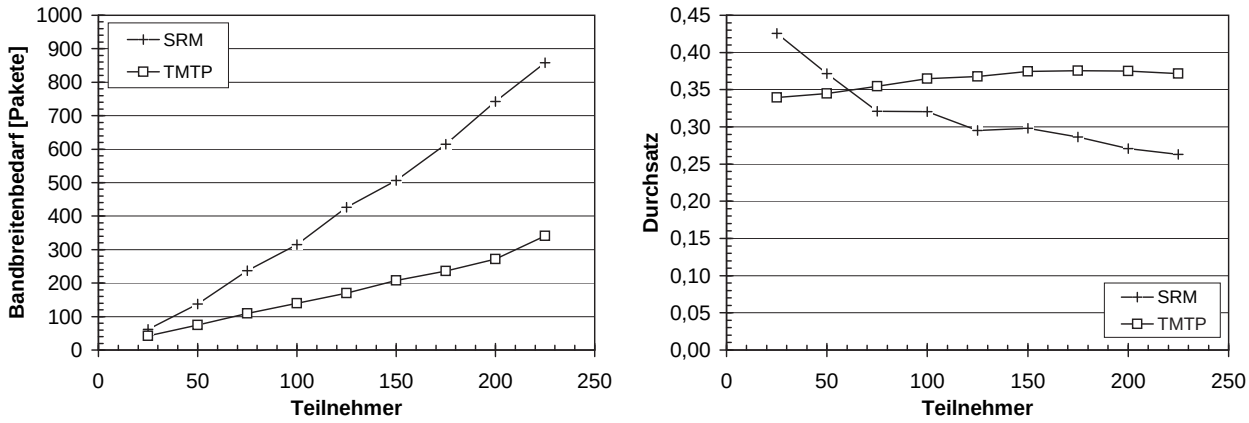



*Abbildung 4.31: Vergleich zwischen einem flachen und einem hierarchischen Protokoll a) Gesamtbandbreitenbedarf und b) Durchsatz*

Empfänger und für die Durchsatzbeschränkung verantwortlich sind. Mit steigender Teilnehmerzahl nimmt der Durchsatz von SRM allerdings stark ab, während er für TMTP bedingt durch die konstante Größe lokaler Gruppen ebenfalls in etwa konstant bleibt.

#### 4.7.2.5 Zusammenfassung

Zusammenfassend lässt sich für alle drei untersuchten Protokollklassen eine sehr gute Übereinstimmung zwischen Analyse– und Simulationsresultaten bescheinigen. Das vorausgesagte qualitative Verhalten konnte durch die Simulation bestätigt werden und in den meisten Fällen zeigte auch der quantitative Vergleich lediglich geringe Abweichungen. Aufgrund der geringen praktischen Bedeutung von flachen senderinitiierten Protokollen wurde auf eine Darstellung und Diskussion vorhandener Simulationsergebnisse zur Klasse K-ACK verzichtet. Auch hier konnten durchgeführte Simulationen die Analyseergebnisse sowohl qualitativ als auch quantitativ bestätigen. Abschließend sei nochmals darauf hingewiesen, dass es sich bei den simulierten Protokollen um möglichst realitätstreue Implementierungen existierender Protokolle handelt und nicht um Nachbildungen der analysierten Protokollklassen. Dies ist von großer Wichtigkeit, da sonst die Gefahr nicht ausgeschlossen werden könnte, potentiell vorhandene Fehler in den Modellannahmen der Analyse auch auf die Simulationen zu übertragen, was deren Aussage– und Beweiskraft wesentlich geschmälert hätte. Die hier simulierten realitätstreuen Protokollimplementierungen besitzen dagegen eine hohe Beweiskraft.

## 4.8 Diskussion

### 4.8.1 Zusammenfassung der Analyse– und Simulationsergebnisse

In diesem Kapitel wurde eine umfassende Analyse von zuverlässigen Multicast-Transportprotokollen präsentiert. Aus den verschiedenen Protokollklassen einen eindeutigen Favoriten zu bestimmen ist schwierig, da die Fehlersemantiken der Protokollklassen unterschiedlich sind und nicht eine einzige Protokollklasse bei allen Ergebnissen die besten Resultate zeigte.

Die Ergebnisse lassen sich folgendermaßen zusammenfassen. K-ACK bietet keine Skalierbarkeit für große Gruppen und hohe Senderaten. Für kleine Gruppen und niedrige Senderaten bietet K-ACK aber niedrige Verzögerungen. K-NACK bietet bereits eine deutlich verbesserte Skalierbarkeit gegenüber K-ACK, da negative Quittungen zu einer geringeren Anzahl notwendiger Quittungen führen. Bezüglich des Gesamtbandbreitenbedarfs liegt K-NACK in der Klasse der flachen Protokolle am niedrigsten. K-NAPP bietet den besten Durchsatz flacher Protokollklassen, der bei niedriger Nachrichtenfehlerrate durchaus für viele Anwendungen ausreichend sein könnte. Dem stehen mehrere Nachteile gegenüber. Ein Nachteil ergibt sich durch die starke Abhängigkeit von der Nachrichtenfehlerrate empfängerinitiierter Protokolle allgemein und durch die Multicast-basierte Quittungsunterdrückung von K-NAPP im Besonderen. So ist die Skalierbarkeit nur gegeben, solange die Nachrichtenfehlerrate niedrig liegt, da ansonsten zu viele negative Quittungen die Skalierbarkeit beschränken. Dieser Effekt kann bereits durch wenige Empfänger mit hoher Nachrichtenfehlerrate ausgelöst werden. Ungewöhnlich erscheint der trotz des besten Durchsatzes flacher Protokolle höchste Gesamtbandbreitenbedarf von K-NAPP. Die Multicast-basierte Quittungsunterdrückung entlastet zwar den Sender, was den Durchsatz verbessert, bürdet aber allen Empfängern zusätzliche Last auf, was den Gesamtbandbreitenbedarf steigert. Schließlich führt K-NAPP zu den höchsten Nachrichtenverzögerungen. Zum einen erhöht die empfängerinitiierte Fehlererkennung generell die Verzögerung für Übertragungswiederholungen, zum anderen muss eine zusätzliche, zufällige Wartezeit akzeptiert werden, da ansonsten die Quittungsunterdrückung nicht effizient arbeitet.

Bei den hierarchischen Protokollen bietet K-HACK den schlechtesten Durchsatz, kann aber mit den niedrigsten Verzögerungen überzeugen. K-HNAPP bietet dagegen den besten Durchsatz, aber auch die höchsten Verzögerungen. Bezüglich des Gesamtbandbreitenbedarfs ergeben sich keine entscheidenden Unterschiede, wenn auch, bedingt durch die kleinen Mitgliederzahlen lokaler Gruppen, K-HNAPP hier besser abschneidet als K-HACK. Werden aggregierte Quittungen in die Überlegungen einbezogen ergeben sich zusätzliche Nachrichten, die den Durchsatz verschlechtern und den Gesamtbandbreitenbedarf erhöhen. Die Unterschiede sind allerdings gering und unter der Verwendung von periodischen aggregierten Quittungen wie bei Klasse K-HANAPP vernachlässigbar.

Hierarchische Protokolle zeigen eine unmittelbare Abhängigkeit der Ergebnisse bezüglich der Bandbreite und der CPU-Belastung von der Lokalgruppenüberlappung. Ist diese hoch, besitzen auch hierarchische Protokolle eine eingeschränkte Skalierbarkeit. Generell ist es dann vorteilhaft, größere lokale Gruppen und damit weniger lokale Gruppen einzurichten, um die Skalierbarkeit zu verbessern. Einen Sonderfall nimmt die Klasse K-HAACK ein, da Übertragungswiederholungen mit Unicast gesendet werden können. Werden Übertragungswiederholungen mit Unicast gesendet, ist keine Abhängigkeit von der Lokalgruppenüberlappung vorhanden.

Generell zeigen alle Ergebnisse die Überlegenheit hierarchischer Protokolle. Dies wurde aufgrund einzelner analytischer Ergebnisse zwar schon zuvor, beispielsweise von [Levine & Garcia-Luna-Aceves 1998], festgestellt, jedoch nicht in dieser Vollständigkeit mit einer Bandbreiten–, einer CPU– und einer Verzögerungsbetrachtung und nicht mit einem derart exakten Systemmodell. Hierarchische Protokolle zeigen bei einer geringen Lokalgruppenüberlappung oder bei der Verwendung von Unicast für die Übertragungswiederholungen unbegrenzte Skalierbarkeit bezüglich der Teilnehmerzahl, höhere Durchsatzraten, geringeren Bandbreitenbedarf und geringere Verzögerungen im Vergleich zu flachen Protokollklassen. Aufgrund dieser Ergebnisse ist der Einsatz hierarchischer Verfahren für Multicast-Übertragungen an große Gruppen dringend zu empfehlen. Eine Zusammenfassung der Ergebnisse ist in Tabelle 4.2 gegeben.

Die aufgeführten Simulationsergebnisse der Transportprotokolle SRM, RMTP und TMTP haben die Analyseergebnisse untermauert. Sie lassen sich wie folgt zusammenfassen. Die Analyseergebnisse bezüglich der Unterschiede zwischen senderinitiierten und empfängerinitiierten Protokollen haben sich bestätigt. Die Eigenschaften empfängerinitiierter Protokolle besitzen eine proportionale Abhängigkeit von der Senderate, aufgrund dessen, dass eine verlorene Nachricht mit der ersten erfolgreich empfangenen Folgenachricht detektiert wird. Dies erhöht bei niedriger Senderate die Verzögerungszeiten gegenüber senderinitiierten Protokollen erheblich. Der Vorteil hierarchischer Protokolle konnte in den Simulationen ebenfalls bestätigt werden, obwohl die Teilnehmerzahl mit bis zu 250 Empfän-

| Transportprotokollklasse | Bandbreitenbedarf | Durchsatz | Verzögerung | Skalierbarkeit |
|---|---|---|---|---|
| K-ACK | ⊖ | ⊖⊖ | ⊕ | ⊖⊖ |
| K-NACK | ⊙ | ⊖ | ⊖ | ⊖ |
| K-NAPP | ⊖⊖ | ⊙ | ⊖⊖ | ⊖ |
| K-HACK | ⊕ | ⊕ | ⊕⊕ | ⊕ |
| K-HNAPP | ⊕⊕ | ⊕⊕ | ⊖⊖ | ⊕⊕ |
| K-HAACK | ⊕⊕ | ⊕ | ⊕⊕ | ⊕⊕ |
| K-HANAPP | ⊕⊕ | ⊕⊕ | ⊖⊖ | ⊕⊕ |
| ⊖⊖ sehr schlecht | ⊖ schlecht | ⊙ durchschnittlich | ⊕ gut | ⊕⊕ sehr gut |

*Tabelle 4.2: Zusammenfassung der Ergebnisse zuverlässiger Transportprotokolle*

gern vergleichsweise gering war. Schließlich hat die Simulation unterschiedlicher Verzweigungsgrade die prognostizierten Auswirkungen auf die Protokollleistung bestätigt. Angesichts der großen Übereinstimmung zwischen den simulierten und analysierten Resultaten ist dies eine nachdrückliche Verifizierung der Analyse.

### 4.8.2 Optimaler Verzweigungsgrad und Konsequenzen für den Token-Repository-Service

Mit der Bestätigung der Überlegenheit hierarchischer gegenüber flachen Transportprotokollen haben die Analyse– und Simulationsresultate auch die Notwendigkeit aufgezeigt, den TRS-Dienst zum effizienten Aufbau der Kontrollbäume einzusetzen. Als weiteres wichtiges Argument für den Einsatz von TRS konnte die hohe Abhängigkeit der Leistungen hierarchischer Protokolle vom Verzweigungsgrad des Kontrollbaums nachgewiesen werden. Im Gegensatz zu alternativen Ansätzen erlaubt der TRS-Dienst, insbesondere TRS-M, den Verzweigungsgrad frei zu konfigurieren. Ein an die jeweilige Situation im Netzwerk angepasster Verzweigungsgrad verringert den Nachrichtenaufwand, erhöht den Durchsatz und erzielt eine geringere Umlaufverzögerung für das Multicast-Transportprotokoll.
Aus den Analyse– und Simulationsergebnissen der vorigen Abschnitte lassen sich folgende Erkenntnisse zum optimalen Verzweigungsgrad ableiten, wobei die Begriffe *großer* und *kleiner Verzweigungsgrad* unten genauer definiert werden:

- Abhängigkeit der Nachrichtenverzögerung vom Verzweigungsgrad:

  Die Maximalverzögerung bis zur Auslieferung einer Nachricht an alle Empfänger lässt sich generell mit einem großen Verzweigungsgrad verringern. Dies trifft ebenfalls für die Umlaufverzögerung der Protokollklassen mit aggregierten Quittungen, K-HAACK und K-HANAPP, zu. Für die Protokollklassen ohne aggregierte Quittungen führt ein kleiner Verzweigungsgrad zu niedrigeren Umlaufverzögerungen. Die durchschnittliche Auslieferungsverzögerung wird durch den Verzweigungsgrad kaum beeinflusst.

  Die Maximalverzögerung aller Protokollklassen und die Umlaufverzögerung der Protokollklassen mit aggregierten Quittungen sinkt mit abnehmender Höhe des Kontrollbaums, was einem großen Verzweigungsgrad entspricht. Ohne aggregierte Quittungen ist die Umlaufverzögerung lediglich durch den Sender und seine direkten Kindknoten im Kontrollbaum bestimmt, weshalb bei K-HACK und K-HNAPP ein kleiner Verzweigungsgrad die Umlaufverzögerung verringert.

- Abhängigkeit des Bandbreitenbedarf und der CPU-Belastung vom Verzweigungsgrad:

  Der optimale Verzweigungsgrad wird für den Bandbreitenbedarf und die CPU-Belastung sowie den daraus ermittelten maximalen Durchsatzraten überwiegend von der Lokalgruppenüberlappung bestimmt. Bei kleiner Wahrscheinlichkeit für Lokalgruppenüberlappung führt ein kleiner

Verzweigungsgrad zu optimalen Ergebnissen und entsprechend ist bei großer Wahrscheinlichkeit für Lokalgruppenüberlappung ein großer Verzweigungsgrad vorzuziehen.

Eine hohe Wahrscheinlichkeit für Lokalgruppenüberlappung führt zum Empfang vieler Multicast-Übertragungswiederholungen und –Quittungen anderer lokalen Gruppen, was die Last der Empfänger und Gruppenführer erhöht. Durch einen großen Verzweigungsgrad kann die Anzahl lokaler Gruppen verringert werden und damit die Anzahl empfangener Nachrichten fremder lokaler Gruppen. Bei geringer Lokalgruppenüberlappung ist ein kleiner Verzweigungsgrad vorzuziehen, da damit eine Lastverteilung auf möglichst viele Gruppenführer erfolgt.

Die Protokollklasse K-HAACK stellt eine Ausnahme dar. Bei geeigneter Konfiguration von K-HAACK wird weder für Übertragungswiederholungen noch für die Quittungsmeldungen ein Multicast-Dienst verwendet. Damit ist keine Abhängigkeit mehr von der Lokalgruppenüberlappung gegeben. Für K-HAACK ist daher generell ein kleiner Verzweigungsgrad zur Optimierung von Bandbreitenbedarf und CPU-Belastung anzustreben.

Wenn in den obigen Ausführungen ein *kleiner Verzweigungsgrad* empfohlen wurde, so ist damit in der Praxis der Verzweigungsgrad zwei gemeint. Ein Verzweigungsgrad von eins scheidet aufgrund der hohen Maximalverzögerung (siehe Ergebnisse in Abbildung 4.22) und aufgrund des hohen Reorganisationsaufwands und der geringen Zuverlässigkeit des Kontrollbaums (siehe Ausführungen in Abschnitt 3.5.2) aus. Der Begriff *großer Verzweigungsgrad* lässt sich nicht einer einzelnen Zahl zuordnen. Hier muss die optimale Größe berechnet werden. Neben dem wichtigsten Einflussparameter, der Lokalgruppenüberlappung, sind für exakte Ergebnisse zudem die Nachrichtenverlustwahrscheinlichkeit und die Teilnehmerzahl der Multicast-Gruppe zu bestimmen. Daraus kann der optimale Verzweigungsgrad mit den Formeln aus den Abschnitten 4.5.2-4.5.4 für die Bandbreite $E(W)$, die CPU-Belastung $E(C)$ und der Verzögerung $E(S)$ durch eine Minimalwertbestimmung berechnet werden.

Wird der so ermittelte Verzweigungsgrad unterschritten, so wird nicht die optimale Protokollleistung erzielt. Eine Überschreitung führt ebenfalls zu einer Minderleistung, im ungünstigsten Fall ist überdies damit zu rechnen, dass eine Überschreitung den Gruppenführer einer lokalen Gruppen überlastet und damit die Protokollleistung stark einbricht (siehe beispielsweise Abbildung 4.22b).

Alle drei Realisierungsvarianten von TRS ermöglichen es, flexibel konfigurierte und deutlich größere Verzweigungsgrade als ERS zu erzielen und damit in vielen Fällen eine Leistungsoptimierung der Transportprotokolle vorzunehmen. Mit TRS-M können Kontrollbäume mit exakt dem vorgegebenen Verzweigungsgrad erstellt werden, so dass die Berechnung des optimalen Verzweigungsgrads unmittelbar zur Konfiguration übernommen werden kann. Der Einsatz des TRS-Diensts ist abschließend nicht nur aufgrund dessen besserer Skalierbarkeit gegenüber herkömmlichen Verfahren wie ERS und ERA zu empfehlen, sondern auch aufgrund der Möglichkeit, Kontrollbäume mit flexibel konfigurierten Verzweigungsgraden zu erstellen und damit die Leistung der Transportprotokolle zu optimieren.

# Kapitel 5

# Resümee

## 5.1 Zusammenfassung

Hierarchische Transportprotokolle erlauben die skalierbare Realisierung eines zuverlässigen Multicast-Diensts durch die Verteilung der Last für Übertragungswiederholungen im Kontrollbaum. Der Kontrollbaum besteht aus den Empfängern der Multicast-Gruppe. Gruppenführer innerhalb des Kontrollbaums übernehmen selbständig Übertragungswiederholungen an Teilmengen der Empfänger, was zu einer Entlastung des Senders führt. Einem Einsatz hierarchischer Transportprotokolle geht der notwendige Aufbau des Kontrollbaums voraus.

Der erste Schwerpunkt dieser Arbeit widmete sich dem Kontrollbaumaufbau. Mit dem Token-Repository-Service (TRS) wurde ein neues Verfahren mit entscheidenden Vorteilen gegenüber herkömmlichen Ansätzen vorgestellt. Der TRS-Dienst verwaltet die notwendige Information zum Kontrollbaumaufbau in sogenannten Marken. Eine Marke repräsentiert das Recht für einen neuen Teilnehmer, sich an einen bestimmten Gruppenführer im Kontrollbaum anzumelden. Die Marken werden auf Servern, genannt RepServern, gespeichert. Die effiziente Verwaltung der Marken ist Aufgabe der drei vorgestellten Realisierungsvarianten TRS-R, TRS-K und TRS-M. TRS-R basiert auf einer erweiternden Ringsuche zwischen den RepServern. Dieses Verfahren benötigt die geringste Infrastruktur, ist aber bezüglich des Nachrichtenaufwands, der Verzögerung im Kontrollbaum, der Kontrollbaumhöhe und der Abhängigkeit vom Routing-Protokoll oftmals im Nachteil. Dagegen organisiert TRS-K alle RepServer in einer Hierarchie. Dies erlaubt es auf die erweiternde Ringsuche zu verzichten, weshalb TRS-K durch den geringsten Nachrichtenaufwand hervorragende Skalierbarkeit erreicht. TRS-M ist schließlich eine Erweiterung von TRS-K, mit deren Hilfe Metriken bei der Auswahl des Gruppenführers im Kontrollbaum berücksichtigt werden können. Eine mögliche Anwendung der Metriken ist die Erstellung von Kontrollbäumen mit minimaler Höhe, was deren Zuverlässigkeit maximiert und geringere Nachrichtenverzögerungen erzielt. Berechnungen und Simulationen haben gezeigt, dass alle drei

Realisierungsvarianten des Token-Repository-Service gegenüber bisherigen Verfahren eine exzellente Skalierbarkeit erzielen. Aufgrund der Unabhängigkeit des TRS-Diensts vom Grad der Multicast-Unterstützung und des Multicast-Routing-Protokolls im Netzwerk ist TRS im Gegensatz zu bisherigen Verfahren auch in unidirektionalen Multicast-Netzen und mit Routing-Protokollen, die nur einen Sender erlauben, anwendbar. Eine prototypische Implementierung des TRS-Diensts ist in [Walz 1999] beschrieben.

Der zweite Schwerpunkt der vorliegenden Arbeit war die ausführliche Analyse zuverlässiger Transportprotokolle. Ausgangspunkt der Analyse war dabei nicht die Untersuchung konkreter Protokolle — deren große Anzahl und zum Teil sehr speziellen Eigenschaften und Optimierungen für einzelne Anwendungen sprachen dagegen — sondern die Identifizierung wichtiger generischer Protokollklassen und deren anschließende Beurteilung. So gelang es, allgemeingültige Aussagen zu erzielen, die eine Beurteilung oder Auswahl vorgegebener Protokolle für einen konkreten Einsatzzweck erleichtern. Beispielsweise haben sich bezüglich der Verzögerungen entscheidende Unterschiede gezeigt. Sind geringe Verzögerungen der Nachrichtenauslieferung von hohem Interesse, so können nur senderinitiierte Protokolle eine generelle Empfehlung erhalten. Die Fehlerdetektion wird vom Sender aufgrund ausbleibender positiver Quittungen durchgeführt, was eine schnelle Erkennung und rasche Übertragungswiederholungen ermöglicht. Dagegen können empfängerinitiierte Protokolle einen Nachrichtenverlust erst mit einer nachfolgenden und erfolgreich empfangenen Nachricht bei einem Empfänger erkennen. Empfängerinitiierte Protokolle verwenden keine oder lediglich periodische positive Quittungen. Stattdessen wird ein Nachrichtenverlust vom Empfänger mit einer negativen Quittung an den Sender angezeigt. Vor allem bei geringen Senderaten verzögert dies Übertragungswiederholungen erheblich. Senderinitiierte Protokolle sind allerdings keineswegs generell im Vorteil. Deren große Schwäche ist ein hoher Nachrichtenaufwand durch die Vielzahl positiver Bestätigungen. Deshalb können empfängerinitiierte Protokolle mit einer besseren Skalierbarkeit große Gruppen und hohe Senderaten besser bewältigen.

Für große Gruppen ab ungefähr 1000 Empfängern ist der Einsatz hierarchischer Protokolle empfehlenswert. Diese bieten bei einer geringen Lokalgruppenüberlappung unbeschränkte Skalierbarkeit. Bei einer erhöhten Lokalgruppenüberlappung verschlechtern sich die Ergebnisse für hierarchische Verfahren. Allerdings kann mit einer Anpassung des Verzweigungsgrads, wie es der TRS-Dienst erlaubt, dem entgegengewirkt werden. Generell sollte einer höherer Lokalgruppenüberlappung ebenso mit einem höheren Verzweigungsgrad begegnet werden, um die Skalierbarkeit, den Durchsatz, den Bandbreitenverbrauch und die Verzögerung zu optimieren. Bei einer zu großen Lokalgruppenüberlappung kann die unbeschränkte Skalierbarkeit durch Übertragungswiederholungen mittels Unicast gesichert werden. Hierarchische Verfahren bieten allerdings nicht nur unbeschränkte Skalierbarkeit, auch bezüglich des verursachten Nachrichtenaufwands, der erzielbaren Nachrichtenverzögerung und des Durchsatzes sind sie durch eine Verteilung der Last gegenüber nicht hierarchischen Protokollen

im Vorteil. Die untersuchten hierarchischen Protokollklassen mit aggregierten Quittungen können zudem weitreichendere Zuverlässigkeitsgarantien geben, da selbst der Ausfall eines Empfängers nicht zum Nachrichtenverlust für Knoten führt. Die Ergebnisse haben gezeigt, dass aggregierte Quittungen einen lediglich moderaten Zusatzaufwand verursachen, sodass deren Einsatz lohnend erscheint. Abschließend lässt sich durch die dargelegten Analyse– und Simulationsergebnisse die Überlegenheit hierarchischer Transportprotokolle bestätigen und damit die Notwendigkeit für den TRS-Dienst belegen.

## 5.2 Bewertung und Ausblick

Im Gegensatz zu dem großen Forschungsinteresse an Multicast-Kommunikation findet die praktische Verbreitung vergleichsweise langsam statt und hinkt dem Stand der Forschung weit hinterher. Bei Betrachtung der notwendigen Änderungen ist die langsame Einführung für einen Multicast-Dienst allerdings nicht ungewöhnlich. Die technischen und sicherlich auch bürokratischen Hürden, neue Routingsoftware in einem derart großen System wie dem Internet zu etablieren, wird häufig unterschätzt. Zudem wird ohne auf Multicast zugeschnittene Anwendungen der Bedarf von den Netzbetreibern nicht erkannt und Multicast nicht verfügbar gemacht. Ohne großflächige Multicast-Unterstützung im Internet werden jedoch kaum derartige Anwendungen entstehen können. Das neue und vereinfachte Routing-Protokoll „senderspezifisches PIM“ sollte genügend Anreize für Netzbetreiber schaffen, nun für eine zügige Multicast-Unterstützung zu sorgen. Die Hoffnungen sind durchaus berechtigt, haben doch viele Netzbetreiber ihre Zustimmung signalisiert.

Der Multicast-Unterstützung in der Vermittlungsschicht müssen Transportprotokolle folgen, um Anwendungen für die Endbenutzer zu ermöglichen. Aufgrund der unterschiedlichsten Anforderungen, wie zuverlässige oder unzuverlässige Kommunikation, große oder kleine Teilnehmergruppen, um nur die wichtigsten zu nennen, erscheint die Etablierung mehrerer Transportprotokolle unausweichlich. Die vorliegende Arbeit trägt hier entscheidend für eine geeignete Auswahl bei. Vorstellbar wäre, dass weiterführende Arbeiten nicht nur die Mechanismen zur Sicherstellung der Zuverlässigkeit untersuchen, sondern ebenfalls umfassend Fluss– und Überlastkontrolle sowie die wichtige Wechselwirkung der Protokolle mit TCP untersuchen.

Speziell für große Gruppen, welche zwingend den Einsatz hierarchischer Transportprotokolle zur zuverlässigen Kommunikation erfordern, leistet die vorliegende Arbeit einen weiteren entscheidenden Beitrag. Bisherigen Verfahren zum Aufbau des Kontrollbaums wurde mangelnde Skalierbarkeit, Flexibilität und Universalität nachgewiesen und mit dem Token-Repository-Service ein neues Verfahren vorgestellt, das diese Nachteile behebt. Die erforderliche Infrastruktur kann zu Beginn aus wenigen Rechnern bestehen, deren Anzahl sukzessive, entsprechend dem zunehmenden Bedarf an zuverlässiger Multicast-Kommunikation, erhöht wird. Denkbar und wünschenswert wäre, dieses Verfahren

und die nun etablierte Infrastruktur nicht nur für den Aufbau der Hierarchie einzusetzen, sondern auch Multicast-Adressen damit zu verwalten, statistische Daten über die Gruppen zu sammeln (z.B. Gruppengröße, geographische Ausdehnung, Lokalgruppenüberlappung) oder auch Abrechnungsverfahren zu unterstützen. Die mögliche Eruierung derartiger Erweiterungen wird zukünftigen Arbeiten vorbehalten bleiben.

# Anhang A

# Analyse der verbleibenden Multicast-Protokolle

## A.1 Übersicht

Im Hauptteil der Arbeit wurden nicht alle Protokollklassen analysiert, da sich deren Berechnung lediglich im Detail von zuvor erfolgten Berechnungen unterscheidet und keine gänzlich neuen Methoden bedingt. Dazu gehören für alle Analysen die Protokolle mit aggregierten Quittungen, K-HAACK und K-HANAPP. Zudem wurde die Analyse der CPU-Belastung exemplarisch einzig an der Protokollklasse K-ACK durchgeführt, da die Berechnungsmethode ähnlich zur Bandbreitenanalyse ist. Diese bisher ausgesparten Berechnungen werden im Folgenden dargelegt. Die numerischen Auswertungen zu allen Protokollklassen finden sich in Abschnitt 4.6.

## A.2 Analyse

Alle folgenden Analysen basieren auf den Berechnungen der Übertragungswiederholungen und der Anzahl von Quittungen aus Abschnitt 4.5.1. Außerdem sei auf die Erläuterungen zu den Analysen aus den Abschnitten 4.5.2-4.5.4 verwiesen.

### A.2.1 Analyse der Bandbreite

#### A.2.1.1 Hierarchisches senderinitiiertes Protokoll mit aggregierten Quittungen (K-HAACK)

Zur einfacheren Berechnung wird der Protokollablauf der Klasse K-HAACK in zwei Phasen aufgeteilt. In der ersten Phase werden die Daten übertragen, wiederholt und alle positiven Quittungen

gesammelt, bis diese von allen Mitgliedern einer lokalen Gruppe empfangen wurden. In der zweiten Phase werden die noch fehlenden aggregierten Quittungen nachgefordert, die in der ersten Phase noch nicht eintrafen. Zu beachten ist, dass viele aggregierte Quittungen bereits in der ersten Phase anstelle von positiven Quittungen gesendet werden, da Gruppenführer, nachdem alle Bedingungen für das Senden von aggregierten Quittungen erfüllt sind, nur noch diese versenden (vgl. Abschnitt 4.2.4).

**Sender:** Der Bandbreitenbedarf des Senders ergibt sich aus dem Senden von Multicast-, $M_m^{HAACK}$, und Unicast-Übertragungen, $M_u^{HAACK}$, dem Empfangen von positiven Quittungen, $\widetilde{L}_a^{HAACK}$, dem Nachfordern von aggregierten Quittungen, $L_{aaq}^{HAACK}$, und schließlich dem Empfangen von aggregierten Quittungen, $\widetilde{L}_{aa}^{HAACK}$. Die Berechnungen dieser Größen kann aus Abschnitt 4.5.1.6 entnommen werden. Der gesamte Bandbreitenbedarf ist:

$$\begin{aligned} E(W_S^{HAACK}) &= E(M_u^{HAACK})N_u E(W_{d,u}) + E(M_m^{HAACK})E(W_{d,m}) \\ &\quad + E(\widetilde{L}_a^{HAACK})E(W_a) + E(L_{aaq}^{HAACK})E(W_{aaq}) + E(\widetilde{L}_{aa}^{HAACK})E(W_{aa}). \end{aligned} \tag{A.1}$$

$W_{d,m}$ und $W_{d,u}$ sind der Bandbreitenbedarf für eine Multicast- bzw. Unicast-Übertragung. $W_a$ und $W_{aa}$ sind der Bandbreitenbedarf für eine positive respektive aggregierte Quittung und schließlich ist $W_{aaq}$ der Bandbreitenbedarf für eine Quittungsnachforderung, die mittels Unicast an die betreffenden Kindknoten gesendet wird. Die Berechnung der Anzahl von Unicast-Nachrichten pro Übertragungswiederholung, $N_u$, ist der Formel 4.60 zu entnehmen.

**Empfänger (ohne Gruppenführer):** Bei einem Empfänger läuft der folgende Bandbreitenbedarf auf:

$$\begin{aligned} E(W_R^{HAACK}) &= E(\widetilde{N}_{r,t}^{HAACK})E(W_d) + E(L_a^{HAACK})E(W_a) + E(L_{aa}^{HAACK})E(W_{aa}) \\ &\quad + E(\widetilde{L}_{aaq}^{HAACK})\Big(E(W_{aa}) + E(W_{aaq})\Big). \end{aligned} \tag{A.2}$$

Ein Empfänger erhält nicht nur die Übertragungswiederholungen des Vaterknotens, sondern auch Übertragungswiederholungen von Gruppenführern anderer lokaler Gruppen, die in $\widetilde{N}_{r,t}^{HAACK}$ berücksichtigt sind (siehe Formel 4.62).

**Gruppenführer (ohne Sender):** Wiederum müssen sowohl der Bandbreitenbedarf des Senders als auch der des Empfängers berücksichtigt werden:

$$E(W_G^{HAACK}) = E(W_S^{HAACK}) + E(W_R^{HAACK}) - E(W_{d,m}(1)) - \Big(E(M^{HAACK}) - 1\Big)(1 - q_D)p_l E(W_d). \tag{A.3}$$

Der Gesamtdurchsatz und der Gesamtbandbreitenbedarf werden äquivalent zur Protokollklasse K-HACK berechnet.

#### A.2.1.2 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung und aggregierten Quittungen (K-HANAPP)

Analog zur vorigen Protokollklasse wird der Protokollablauf in zwei Phasen aufgeteilt: Senden aller notwendigen Übertragungen und Nachfordern aller ausgebliebenen aggregierten Quittungen. In einem empfängerinitiierten Protokoll machen aggregierte Quittungen nur dann Sinn, wenn diese periodisch gesendet werden, da sonst der Aufwand vergleichbar mit dem für senderinitiierte Protokolle ist. Deshalb wird später in den numerischen Auswertungen der Bandbreitenbedarf für aggregierte Quittungen, $W_{aa,\phi}$, und Nachforderungen für aggregierte Quittungen, $W_{aaq,\phi}$, auf entsprechende Teilbeträge einer gesamten Quittung gesetzt.

**Sender:** Der Bandbreitenbedarf des Sender ist:

$$\begin{aligned} E(W_S^{HANAPP}) &= E(M^{HANAPP})E(W_d) + E(\widetilde{L}^{HANAPP})E(W_n) \\ &\quad + E(L_{aaq}^{HANAPP})E(W_{aaq,\phi}) + E(\widetilde{L}_{aa}^{HANAPP})E(W_{aa,\phi}). \end{aligned} \tag{A.4}$$

Die Größen $E(M^{HANAPP})$, $E(\widetilde{L}^{HANAPP})$, $E(L_{aaq}^{HANAPP})$ und $E(\widetilde{L}_{aa}^{HANAPP})$ wurden bereits in Abschnitt 4.5.1.7 bestimmt.

**Empfänger (ohne Gruppenführer):** Der Bandbreitenbedarf eines Empfängers ist:

$$\begin{aligned} E(W_R^{HANAPP}) &= E(M^{HANAPP})(1-q_D)E(W_d) \\ &\quad + \Big(E(O_r^{HANAPP})-1\Big)\frac{E(L_\phi^{HANAPP})}{E(R_N)}E(W_n) + E(W_{aa,\phi}) + E(\widetilde{L}_{aaq}^{HANAPP})\Big(E(W_{aaq,\phi}) + E(W_{aa,\phi})\Big) \\ &\quad \underbrace{+\Big[\Big(E(O^{HANAPP})-1\Big)E(L_\phi^{HANAPP}) - \Big(E(O_r^{HANAPP})-1\Big)\frac{E(L_\phi^{HANAPP})}{E(R_N)}\Big](1-q_N)E(W_n)}_{\textit{eigene lokale Gruppe}} \\ &\quad \underbrace{+(G-1)p_l\Big[\Big(E(M^{HANAPP})-1\Big)(1-q_D)E(W_d) + \Big(E(O^{HANAPP})-1\Big)E(L_\phi^{HANAPP})(1-q_N)E(W_n)\Big]}_{\textit{fremde lokale Gruppen}}. \end{aligned} \tag{A.5}$$

Für die Erklärung zu den Berechnungen wird auf die Protokollklasse K-HNAPP verwiesen.

**Gruppenführer (ohne Sender):** Die Berechnung des Bandbreitenbedarfs des Gruppenführers ist äquivalent zu den vorigen Klassen:

$$\begin{aligned} E(W_G^{HANAPP}) &= E(W_S^{HANAPP}) + E(W_R^{HANAPP}) - E(W_d(1)) \\ &\quad - p_l\Big[\Big(E(M^{HANAPP})-1\Big)(1-q_D)E(W_d) + \Big(E(O^{HANAPP})-1\Big)E(L_\phi^{HANAPP})(1-q_N)E(W_n)\Big]. \end{aligned} \tag{A.6}$$

Der Gesamtdurchsatz und der Gesamtbandbreitenbedarf werden äquivalent zur Protokollklasse K-HACK berechnet.

## A.2.2 Analyse der CPU-Belastung

In den nachfolgenden Abschnitten wird die bisher nur an K-ACK beispielhaft durchgeführte Analyse der CPU-Belastung aus Abschnitt 4.5.3 auf alle Protokollklassen ausgedehnt.

### A.2.2.1 Empfängerinitiiertes Protokoll (K-NACK)

In einem empfängerinitiierten Protokoll müssen neben dem Sender auch die Empfänger einen Zeitgeber verwalten. Der Zeitgeber des Senders bestimmt die Wartedauer auf ankommende negative Quittungen. Der Zeitgeber eines Empfängers ist notwendig, um nach einer angemessenen Wartezeit auf eine Übertragungswiederholung eine gesendete negative Quittung als verloren anzusehen und eine weitere zu senden. Ansonsten hat die Abbildung 4.6 auch hier Gültigkeit. Die CPU-Belastung des Senders ist:

$$\begin{aligned} C_S^{NACK} &= \textit{(höhere Protokollschicht)} + \textit{(Übertragungen senden)} \\ &\quad + \textit{(Quittungen empfangen)} + \textit{(Zeitgeber)} \\ E(C_S^{NACK}) &= E(X_f) + E(M^{NACK})E(X_d) + E(\widetilde{L}^{NACK})E(X_n) + E(O^{NACK})E(X_t). \end{aligned} \tag{A.7}$$

Die Größen $E(M^{NACK})$, $E(\widetilde{L}^{NACK})$ und $E(O^{NACK})$ wurden in den Formeln 4.20, 4.31 und 4.24 bestimmt. Für einen Empfänger ergibt sich folgende CPU-Belastung:

$$\begin{aligned} E(C_R^{NACK}) &= E(Y_f) + E(M^{NACK})(1-q_D)E(Y_d) \\ &\quad + \Big(E(O_r^{NACK}) - 1\Big)E(Y_n) + \Big(E(O_r^{NACK}) - 2\Big)E(Y_t). \end{aligned} \tag{A.8}$$

Zu beachten ist, dass die letzte, erfolgreiche Übertragung nicht mit einer negativen Quittung beantwortet wird. Ob eine negative Quittung erfolgreich war und zu einer Übertragungswiederholung geführt hat, wird durch das erfolgreiche Empfangen der Übertragungswiederholung festgestellt. Alle nicht erfolgreichen negativen Quittungen führen zum Setzen eines neuen Zeitgebers. Die maximalen Durchsatzraten können analog zur Protokollklasse K-ACK bestimmt werden.

### A.2.2.2 Empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-NAPP)

Analog zur Protokollklasse K-NACK ist die CPU-Belastung des Senders:

$$E(C_S^{NAPP}) = E(X_f) + E(M^{NAPP})E(X_d) + E(\widetilde{L}^{NAPP})E(X_n) + E(O^{NAPP})E(X_t). \tag{A.9}$$

Die Größen $E(M^{NAPP})$, $E(\widetilde{L}^{NAPP})$ und $E(O^{NAPP})$ wurden in Abschnitt 4.5.1.3 ermittelt.

Die CPU-Belastung des Empfängers ergibt sich zu:

$$\begin{aligned} E(C_R^{NAPP}) &= E(Y_f)+E(M^{NAPP})(1-q_D)E(Y_d)+\underbrace{\Big(E(O_r^{NAPP})-1\Big)\frac{E(L_\phi^{NAPP})}{E(R_N)}E(Y_n)}_{\textit{gesendete Quittungen}} \\ &\quad+\underbrace{\Big[\Big(E(O^{NAPP})-1\Big)E(L_\phi^{NAPP})-\Big(E(O_r^{NAPP})-1\Big)\frac{E(L_\phi^{NAPP})}{E(R_N)}\Big](1-q_N)E(Y_n)}_{\textit{empfangene Quittungen}} \\ &\quad+\underbrace{\Big(E(O_r^{NAPP})-2\Big)E(Y_t)}_{\textit{Zeitgeber}}. \end{aligned} \tag{A.10}$$

$L_\phi^{NAPP}$, die durchschnittliche Anzahl gesendeter negativer Quittungen pro Runde, wird entsprechend Formel 4.43 ermittelt und $R_N$, die durchschnittliche Anzahl erfolgloser Empfänger, ist in Formel 4.44 gegeben. Die maximalen Durchsatzraten können analog zur vorigen Protokollklasse bestimmt werden.

### A.2.2.3 Hierarchisches senderinitiiertes Protokoll (K-HACK)

**Sender:** Die Bestimmung der CPU-Belastung beim Sender erfolgt analog zur Protokollklasse K-ACK, da alle Besonderheiten der hierarchischen Organisation in den Größen $E(M^{HACK})$ und $E(\widetilde{L}^{HACK})$ bereits berücksichtigt sind (siehe Abschnitt 4.5.1.1):

$$E(C_S^{HACK}) = E(X_f)+E(M^{HACK})E(X_d)+\Big(E(M^{HACK})-1\Big)E(X_t)+E(\widetilde{L}^{HACK})E(X_a). \tag{A.11}$$

**Empfänger (ohne Gruppenführer):** Der CPU-Aufwand eines Empfängers, $C_R^{HACK}$, ergibt sich aus den empfangenen Übertragungen, die mit einer positiven Quittung bestätigt werden. Zeitgeber müssen bei einem senderinitiierten Protokoll von den Empfängern nicht verwaltet werden. Der CPU-Aufwand ist deshalb:

$$E(C_R^{HACK}) = E(Y_f)+E(\widetilde{N}_{r,t}^{HACK})E(Y_d)+E(\widetilde{N}_r^{HACK})E(Y_a). \tag{A.12}$$

$E(\widetilde{N}_{r,t}^{HACK})$ ist die Gesamtanzahl von empfangenen Übertragungen sowohl vom Sender und Gruppenführer der eigenen lokalen Gruppe, $E(\widetilde{N}_r^{HACK})$, als auch den anderen Gruppenführern, die mit einer bestimmten Wahrscheinlichkeit Nachrichten in andere lokale Gruppen senden (siehe Formeln 4.47 und 4.48). Mit einer positiven Quittung werden einzig die Übertragungen der eigenen lokalen Gruppe bestätigt.

**Gruppenführer (ohne Sender):** Für einen Gruppenführer ergibt sich sowohl der CPU-Aufwand eines Senders als auch der eines Empfängers, allerdings ohne die initiale Übertragung, an der ein Gruppenführer nicht beteiligt ist und ohne die Übergabe der Daten von einer höheren Protokollschicht.

Desweiteren muss berücksichtigt werden, dass ein Gruppenführer bereits Mitglied in zwei lokalen Gruppen ist, was in $\widetilde{N}_g^{HACK}$ berücksichtigt ist. Der CPU-Aufwand ergibt sich zu:

$$\begin{aligned} E(C_G^{HACK}) &= \Big(E(M^{HACK})-1\Big)\Big(E(X_d)+E(X_t)\Big)+E(\widetilde{L}^{HACK})E(X_a) \\ &\quad +E(\widetilde{N}_g^{HACK})E(Y_d)+E(\widetilde{N}_r^{HACK})E(Y_a)+E(Y_f) \qquad (A.13) \\ &= E(C_S^{HACK})+E(C_R^{HACK})-E(X_f)-E(X_d(1))-\Big(E(M^{HACK})-1\Big)(1-q_D)p_l E(Y_d). \qquad (A.14) \end{aligned}$$

Die maximalen Durchsatzraten für Sender, Empfänger und Gruppenführer ergeben sich zu:

$$\Lambda_S^{HACK}=\frac{1}{E(C_S^{HACK})},\ \Lambda_R^{HACK}=\frac{1}{E(C_R^{HACK})},\ \Lambda_G^{HACK}=\frac{1}{E(C_G^{HACK})}. \qquad (A.15)$$

Der gesamte maximale Systemdurchsatz aufgrund der CPU-Belastung ergibt sich aus dem minimalen Durchsatz von Sender, Empfänger und Gruppenführer:

$$\Lambda^{HACK}=min\{\Lambda_S^{HACK},\Lambda_G^{HACK},\Lambda_R^{HACK}\}. \qquad (A.16)$$

#### A.2.2.4 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung (K-HNAPP)

**Sender:** Beim Sender müssen die Zeitgeber berücksichtigt werden, die in jeder Runde, auch in leeren Runden, in denen alle negativen Quittungen verloren gehen, anfallen:

$$E(C_S^{HNAPP}) = E(X_f)+E(M^{HNAPP})E(X_d)+E(\widetilde{L}^{HNAPP})E(X_n)+E(O^{HNAPP})E(X_t)+E(S)E(X_\Phi). \qquad (A.17)$$

$X_\Phi$ ist die CPU-Belastung, die durch periodische Quittungen verursacht wird. In den numerischen Auswertungen wird $X_\Phi$ auf einen Bruchteil der normalen CPU-Belastung gesetzt. Zu beachten ist, dass bei der Berechnung von $\widetilde{L}^{HNAPP}$ in Formel 4.53 bereits berücksichtigt wurde, dass negative Quittungen zudem von anderen lokalen Gruppen empfangen werden können.

**Empfänger (ohne Gruppenführer):** Die Berechnung der CPU-Belastung des Empfängers ist ähnlich zur Protokollklasse K-NAPP. Neben dem Empfangen von Daten und dem Senden von periodischen und negativen Quittungen, muss das Empfangen von negativen Quittungen von anderen Mitgliedern der lokalen Gruppe und außerhalb berücksichtigt werden:

$$\begin{aligned} E(C_R^{HNAPP}) &= E(Y_f)+E(M^{HNAPP})(1-q_D)E(Y_d)+E(Y_\Phi) \\ &\quad +\Big(E(O_r^{HNAPP})-1\Big)\frac{E(L_\phi^{HNAPP})}{E(R_N)}E(Y_n)+\Big(E(O_r^{HNAPP})-2\Big)E(Y_t) \\ &\quad \underbrace{+\Big[\Big(E(O^{HNAPP})-1\Big)E(L_\phi^{HNAPP})-\Big(E(O_r^{HNAPP})-1\Big)\frac{E(L_\phi^{HNAPP})}{E(R_N)}\Big](1-q_N)E(X_n)}_{\textit{eigene lokale Gruppe}} \\ &\quad \underbrace{+(G-1)p_l\Big[\Big(E(M^{HNAPP})-1\Big)E(Y_d)+\Big(E(O^{HNAPP})-1\Big)E(L_\phi^{HNAPP})E(X_n)\Big]}_{\textit{fremde lokale Gruppen}}. \qquad (A.18) \end{aligned}$$

Die Größen $L_{\phi}^{HNAPP}$ und $R_N$ wurden bereits in Abschnitt 4.5.1.5 bestimmt.

**Gruppenführer (ohne Sender):** Der CPU-Aufwand des Gruppenführers ist:

$$\begin{aligned} E(C_G^{HNAPP}) &= E(C_S^{HNAPP}) + E(C_R^{HNAPP}) - E(X_f) - E(X_d(1)) \\ &\quad - p_l\Big[\Big(E(M^{HNAPP}) - 1\Big)E(Y_d) + \Big(E(O^{HNAPP}) - 1\Big)E(L_{\phi}^{HNAPP})(1-q_N)E(Y_n)\Big]. \end{aligned} \quad \text{(A.19)}$$

Die maximalen Durchsatzraten können analog zur vorigen Protokollklasse bestimmt werden.

### A.2.2.5 Hierarchisches senderinitiiertes Protokoll mit aggregierten Quittungen (K-HAACK)

**Sender:** Im Gegensatz zum Bandbreitenbedarf müssen beim Sender zwei Zeitgeber berücksichtigt werden. Ein Zeitgeber steuert die Übertragungswiederholungen, d.h. bestimmt die Zeitspanne für das Sammeln von positiven Quittungen bevor eine Übertragungswiederholung gesendet wird. Der zweite Zeitgeber steuert das Senden von Nachforderungen für aggregierte Quittungen, d.h. die Zeitspanne für das Sammeln von aggregierten Quittungen bevor eine Quittungsnachforderung gesendet wird. Der CPU-Aufwand ergibt sich zu:

$$\begin{aligned} E(C_S^{HAACK}) &= E(X_f) + E(M_u^{HAACK})N_u E(X_{d,u}) + E(M_m^{HAACK})E(X_{d,m}) \\ &\quad + E(M^{HAACK} - 1)E(X_t) + E(\widetilde{L}_a^{HAACK})E(X_a) + E(O_q^{HAACK})E(X_t) \\ &\quad + E(L_{aaq}^{HAACK})E(X_{aaq}) + E(\widetilde{L}_{aa}^{HAACK})E(X_{aa}). \end{aligned} \quad \text{(A.20)}$$

Die Anzahl der Multicast- und Unicast-Übertragungswiederholungen, $M_m^{HAACK}$ und $M_u^{HAACK}$ sowie die Anzahl der Unicast-Nachrichten pro Unicast-Übertragungswiederholung, $N_u$, wurden in den Formeln 4.56, 4.57 und 4.60 bestimmt. Die restlichen Größen für die Anzahl der empfangenen positiven und aggregierten Quittungen, $\widetilde{L}_a^{HAACK}$ und $\widetilde{L}_{aa}^{HAACK}$, die Quittungsnachforderungen, $L_{aaq}^{HAACK}$ sowie die Anzahl der Nachfragerunden, $O_q^{HAACK}$, wurden ebenfalls in Abschnitt 4.5.1.6 bestimmt.

**Empfänger (ohne Gruppenführer):** Der CPU-Aufwand beim Empfänger ist der CPU-Aufwand von Protokollklasse K-HACK mit den zusätzlichen aggregierten Quittungen:

$$\begin{aligned} E(C_R^{HAACK}) &= E(Y_f) + E(\widetilde{N}_{r,t}^{HAACK})E(Y_d) + E(L_a^{HAACK})E(Y_a) \\ &\quad + E(L_{aa}^{HAACK})E(Y_{aa}) + E(\widetilde{L}_{aaq}^{HAACK})\Big(E(Y_{aa}) + E(Y_{aaq})\Big). \end{aligned} \quad \text{(A.21)}$$

**Gruppenführer:** Der Aufwand des Gruppenführers ist:

$$\begin{aligned} E(C_G^{HAACK}) &= E(C_S^{HAACK}) + E(C_R^{HAACK}) \\ &\quad - E(X_f) - E(X_{d,m}(1)) - \Big(E(M_m^{HAACK}) - 1\Big)p_l(1-q_D)E(Y_d). \end{aligned} \quad \text{(A.22)}$$

Die maximalen Durchsatzraten können analog zur vorigen Protokollklasse bestimmt werden.

#### A.2.2.6 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung und aggregierten Quittungen (K-HANAPP)

**Sender:** Wie in der vorigen Protokollklasse werden zwei Phasen berücksichtigt. In der ersten Phase werden alle Übertragungswiederholungen durchgeführt. In der zweiten Phasen werden die fehlenden aggregierten Quittungen nachgefordert:

$$\begin{aligned} E(C_S^{HANAPP}) &= E(X_f) + E(M^{HANAPP})E(X_d) + E(\widetilde{L}^{HANAPP})E(X_n) + E(O^{HANAPP})E(X_t) \\ &\quad + E(O_q^{HANAPP})E(X_t) + E(L_{aaq}^{HANAPP})E(X_{aaq,\phi}) + E(\widetilde{L}_{aa}^{HANAPP})E(X_{aa,\phi}). \end{aligned} \tag{A.23}$$

**Empfänger (ohne Gruppenführer):** Analog zur Protokollklasse K-HNAPP ergibt sich der CPU-Aufwand des Empfängers zu:

$$\begin{aligned} E(C_R^{HANAPP}) &= E(Y_f) + E(M^{HANAPP})(1-q_D)E(Y_d) \\ &\quad + \Big(E(O_r^{HANAPP}) - 1\Big)\frac{E(L_\phi^{HANAPP})}{E(R_N)}E(Y_n) + \Big(E(O_r^{HANAPP}) - 2\Big)E(Y_t) \\ &\quad + E(Y_{aa,\phi}) + E(\widetilde{L}_{aaq}^{HANAPP})\Big(E(Y_{aaq,\phi}) + E(Y_{aa,\phi})\Big) \\ &\quad \underbrace{+\Big[\Big(E(O^{HANAPP}) - 1\Big)E(L_\phi^{HANAPP}) - \Big(E(O_r^{HANAPP}) - 1\Big)\frac{E(L_\phi^{HANAPP})}{E(R_N)}\Big](1-q_N)E(X_n)}_{\textit{eigene lokale Gruppe}} \\ &\quad \underbrace{+(G-1)p_l\Big[\Big(E(M^{HANAPP}) - 1\Big)(1-q_D)E(Y_d) + \Big(E(O^{HANAPP}) - 1\Big)E(L_\phi^{HANAPP})(1-q_N)E(X_n)\Big]}_{\textit{fremde lokale Gruppen}}. \end{aligned} \tag{A.24}$$

**Gruppenführer (ohne Sender):** Zu berücksichtigen ist wiederum, dass der Aufwand für das Empfangen einer weiteren lokalen Gruppe abgezogen werden muss:

$$\begin{aligned} E(C_G^{HANAPP}) &= E(C_S^{HANAPP}) + E(C_R^{HANAPP}) - E(X_f) - E(X_d(1)) \\ &\quad - p_l\Big[\Big(E(M^{HANAPP}) - 1\Big)(1-q_D)E(Y_d) + \Big(E(O^{HANAPP}) - 1\Big)E(L_\phi^{HANAPP})(1-q_N)E(Y_n)\Big]. \end{aligned} \tag{A.25}$$

Die maximalen Durchsatzraten können analog zur vorigen Protokollklasse bestimmt werden.

### A.2.3 Analyse der Verzögerung

In Abschnitt 4.5.4 wurden bei der Verzögerungsanalyse die beiden Protokollklassen K-HAACK und K-HANAPP ausgespart. Deren vollständige Analyse wird im Folgenden wiedergegeben.

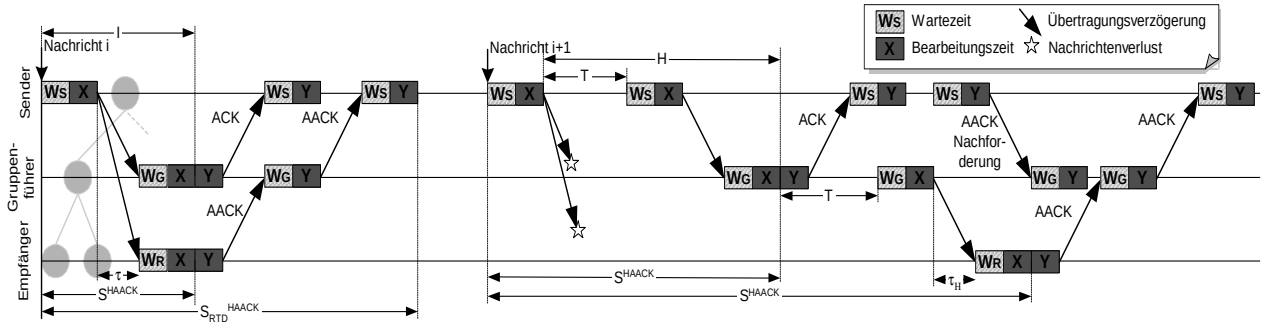



*Abbildung A.1: Verzögerungskomponenten der Protokollklasse K-HAACK*

### A.2.3.1 Hierarchisches senderinitiiertes Protokoll mit aggregierten Quittungen (K-HAACK)

In Abbildung A.1 sind in Ergänzung zu den vorigen Protokollklassen die aggregierten Quittungen enthalten.

**Mittlere Wartezeit einer Nachricht beim Sender:** Bedingt durch die aggregierten Quittungen und der schwellwertbasierten Übertragungswiederholung mittels Unicast oder Multicast lassen sich sechs Nachrichtenflüsse beim Sender unterscheiden:

1. Der initiale Datenübertragungsfluss $\lambda_t^S$ mit Rate $\lambda$ und Bearbeitungszeit $X$.
2. Der Nachrichtenfluss der gesendeten Übertragungswiederholungen mittels Unicast $\lambda_{r,u}^S$. Die Rate beträgt $\lambda E(N_u)E(M_u^{HAACK})$ und die Bearbeitungszeit ist $X$. $E(N_u)$ ist die durchschnittliche Anzahl an Unicast-Nachrichten pro Übertragungsrunde.
3. Die Multicast-Übertragungswiederholungen $\lambda_{r,m}^S$ mit Rate $\lambda(E(M_m^{HAACK})-1)$ und Bearbeitungszeit $X$.
4. Die ankommenden positiven Quittungen $\lambda_a^S$ mit Rate $\lambda E(\widetilde{L}_a^{HAACK})$ und Bearbeitungszeit $Y$.
5. Die ankommenden aggregierten Quittungen $\lambda_{aa}^S$ mit Rate $\lambda E(\widetilde{L}_{aa}^{HAACK})$ und Bearbeitungszeit $Y$.
6. Schließlich der Nachrichtenfluss der Quittungsnachforderungen $\lambda_q^S$ mit Rate $\lambda E(L_{aaq}^{HAACK})$ und Bearbeitungszeit $Y$.

Die Gesamtlast des Senders und die mittlere Nachrichtenwartezeit sind:

$$\begin{aligned}\rho_S^{HAACK} &= \lambda\Big(E(N_u)E(M_u^{HAACK})+E(M_m^{HAACK})\Big)E(X)\\ &\quad+\lambda\Big(E(\widetilde{L}_a^{HAACK})+E(\widetilde{L}_{aa}^{HAACK})+E(L_{aaq}^{HAACK})\Big)E(Y) \qquad \text{(A.26)}\\ E(W_S^{HAACK}) &= \frac{(\lambda_t^S+\lambda_{r,u}^S+\lambda_{r,m}^S)E(X^2)+(\lambda_a^S+\lambda_{aa}^S+\lambda_q^S)E(Y^2)}{2(1-\rho_S^{HAACK})}. \qquad \text{(A.27)}\end{aligned}$$

**Mittlere Wartezeit einer Nachricht beim Empfänger:** Der Empfänger verarbeitet die folgenden drei Nachrichtenflüsse, die gegenüber der Protokollklasse K-HACK um die aggregierten Quittungen erweitert sind:

1. Der Nachrichtenfluss der Übertragungen vom Vaterknoten im Kontrollbaum $\lambda_t^R$. Diese haben die Rate $\lambda E(\widetilde{N}_r^{HAACK})$. Eine erfolgreich empfangene Übertragung führt zum Senden einer positiven oder aggregierten Quittung. Daher ist die Bearbeitungszeit $X+Y$.

2. Der Nachrichtenfluss der Übertragungswiederholungen anderer lokaler Gruppen $\lambda_G^R$ mit Rate $\lambda\Big(E(\widetilde{N}_{r,t}^{HAACK}) - E(\widetilde{N}_r^{HAACK})\Big)$. Die Bearbeitungszeit ist $X$, da keine Quittung gesendet wird.

3. Die Quittungsnachforderungen $\lambda_q^R$ haben die Rate $\lambda E(\widetilde{L}_{aaq}^{HAACK})$. Quittungsnachforderungen führen zum Senden von aggregierten Quittungen, daher beträgt die Bearbeitungszeit $Y+Y$.

Aus den Nachrichtenflüssen folgt die Last des Empfängers und die mittlere Nachrichtenwartezeit bis zur Bearbeitung:

$$\begin{aligned}\rho_R^{HAACK} &= \lambda E(\widetilde{N}_r^{HAACK})E(X+Y) + \lambda\Big(E(\widetilde{N}_{r,t}^{HAACK}) - E(\widetilde{N}_r^{HAACK})\Big)E(X) \\ &\quad + \lambda E(\widetilde{L}_{aaq}^{HAACK})E(Y+Y) \qquad \text{(A.28)} \\ E(W_R^{HAACK}) &= \frac{\lambda_t^R E((X+Y)^2) + \lambda_G^R E(X^2) + \lambda_q^R E((Y+Y)^2)}{2(1-\rho_R^{HAACK})}. \qquad \text{(A.29)}\end{aligned}$$

**Mittlere Wartezeit einer Nachricht beim Gruppenführer:** Die Last und die mittlere Nachrichtenwartezeit des Gruppenführers ergibt sich aus der Lastbetrachtung des Senders ohne die initiale Übertragung und der des Empfängers:

$$\begin{aligned}\rho_G^{HAACK} &= \lambda\Big(N_u E(M_u^{HAACK}) + E(M_m^{HAACK}) - 1\Big)E(X) \\ &\quad + \lambda\Big(E(\widetilde{L}_a^{HAACK}) + E(\widetilde{L}_{aa}^{HAACK}) + E(L_{aaq}^{HAACK})\Big)E(Y) + \lambda E(\widetilde{N}_r^{HAACK})E(X+Y) \\ &\quad + \lambda\Big(E(\widetilde{N}_g^{HAACK}) - E(\widetilde{N}_r^{HAACK})\Big)E(X) + \lambda E(\widetilde{L}_{aaq}^{HAACK})E(Y+Y) \qquad \text{(A.30)} \\ E(W_G^{HAACK}) &= \frac{\Big(\lambda_{r,u}^S + \lambda_{r,m}^S\Big)E(X^2) + \Big(\lambda_a^S + \lambda_{aa}^S + \lambda_q^S\Big)E(Y^2)}{2(1-\rho_G^{HAACK})} \\ &\quad + \frac{\lambda_t^R E((X+Y)^2) + \lambda_G^R E(X^2) + \lambda_q^R E((Y+Y)^2)}{2(1-\rho_G^{HAACK})}. \qquad \text{(A.31)}\end{aligned}$$

**Gesamtverzögerung:** Die initiale Übertragung hat die Verzögerung:

$$E(I) = E(W_S^{HAACK}) + E(X) + \tau + E(W_G^{HAACK}) + E(X). \qquad \text{(A.32)}$$

Eine hierarchische Übertragungswiederholung, unter der Annahme dass der Vaterknoten die Daten erfolgreich empfangen hat und darum eine Übertragungswiederholung durchführen kann, besitzt die Verzögerung:

$$E(H) = \Big(E(M_r^{HAACK})-1\Big)\Big(T+E(W_G^{HAACK})+E(X)\Big)+\tau_H+E(W_G^{HAACK})+E(X). \quad (A.33)$$

Die Berechnung der Durchschnittsverzögerung erfolgt analog zur Protokollklasse K-HACK:

$$E(S_\phi^{HAACK}) = \Big[\sum_{i=o}^{\bar{h}-2} q_D^i(1-q_D)\Big(E(I)+iE(H)\Big)\Big]+q_D^{\bar{h}-1}\Big(E(W_S^{HAACK})+E(X)+(\bar{h}-1)E(H)\Big). \quad (A.34)$$

Für die Bestimmung der Umlaufverzögerung werden aggregierte Quittungen betrachtet, da diese zur Flusskontrolle eingesetzt werden:

$$\begin{aligned} E(S_{RTD}^{HAACK}) = {} & \underbrace{\Big(E(M^{HAACK})-1\Big)\Big(T+E(W_G^{HAACK})+E(X)\Big)}_{\textit{Übertragungen senden}} \\ & + \underbrace{E(X)+E(W_G^{HAACK})+\tau_H+E(W_G^{HAACK})+E(X)}_{\textit{letzte Übertragung empfangen}} \\ & + (h-1)\Big[\underbrace{E(\widetilde{L}_{aaq}^{HAACK})\Big(T+E(W_G^{HAACK})+E(Y)\Big)}_{\textit{Quittungsnachforderungen}} + \underbrace{E(Y)+\tau_H+E(W_G^{HAACK})+E(Y)}_{\textit{letzte aggregierte Quittung}}\Big]. \end{aligned} \quad (A.35)$$

Die Bestimmung der Maximalverzögerung $E(S_\gamma^{HAACK})$ ist analog zur Protokollklasse K-HACK.

### A.2.3.2 Hierarchisches empfängerinitiiertes Protokoll mit Quittungsunterdrückung und aggregierten Quittungen (K-HANAPP)

**Mittlere Wartezeit einer Nachricht beim Sender:** Der Fluss der initialen Datennachrichten $\lambda_t^S$, der negativen Quittungen mit Übertragungswiederholungen $\lambda_r^S$ und der zusätzlichen negativen Quittungen, die keine Übertragungswiederholung auslösen, $\lambda_n^S$, ist analog zur Protokollklasse K-HNAPP. Zusätzlich sind zwei weitere Nachrichtenflüsse zu berücksichtigen. Der Quittungsnachforderungsfluss $\lambda_q^S$ mit Rate $\lambda E(L_{aaq}^{HANAPP})$ und der Fluss der aggregierten Quittungen $\lambda_{aa}^S$ mit Rate $\lambda B$ (siehe Abbildung A.2).

Für beide Flüsse ist eine Bearbeitungszeit von $Z$ angemessen, da sie ähnlich zu den periodischen positiven Quittungen von K-HNAPP nicht nach jeder Datenübertragung gesendet werden. Damit beträgt die Senderlast und die mittlere Wartezeit einer Nachricht bis zur Bearbeitung:

$$\rho_S^{HANAPP} = (\lambda_t^S+\lambda_r^S)E(X)+(\lambda_r^S+\lambda_n^S)E(Y)+(\lambda_q^S+\lambda_{aa}^S)E(Z) \quad (A.36)$$

$$E(W_S^{HANAPP}) = \frac{\lambda_t^S E(X^2)+\lambda_r^S\Big(E(X^2)+E(Y^2)+2E(X)E(Y)\Big)+\lambda_n^S E(Y^2)+(\lambda_{aa}^S+\lambda_q^S)E(Z^2)}{2(1-\rho_S^{HANAPP})}. \quad (A.37)$$

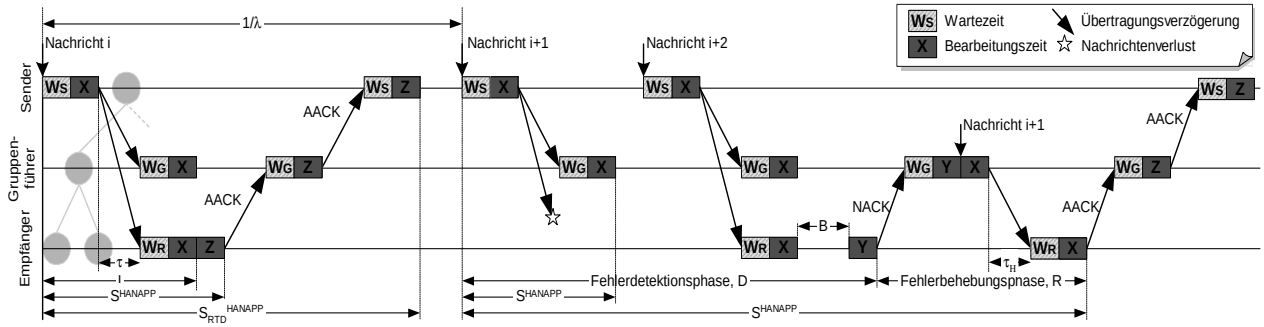



*Abbildung A.2: Verzögerungskomponenten der Protokollklasse K-HANAPP*

**Mittlere Wartezeit einer Nachricht beim Empfänger:** Neben den bereits bekannten Nachrichtenflüssen der Protokollklasse K-HNAPP, Übertragungen $\lambda_t^R$, negative Quittungen senden $\lambda_{n,g}^R$ und empfangen $\lambda_{n,r}^R$ muss noch der Quittungsnachforderungsfluss $\lambda_q^R$ betrachtet werden. $\lambda_q^R$ hat die Rate $\lambda\widetilde{L}_{aaq}^{HANAPP}$ und die Bearbeitungszeit $Z+Z$, da jede Quittungsnachforderung mit einer aggregierten Quittung beantwortet wird. Damit ergibt sich die Last des Empfängers und die mittlere Nachrichtenwartezeit zu:

$$\rho_R^{HANAPP} = \lambda_t^R E(X) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y) + \lambda_q^R E(Z+Z) \tag{A.38}$$

$$E(W_R^{HANAPP}) = \frac{\lambda_t^R E(X^2) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y^2) + \lambda_q^R E((Z+Z)^2)}{2(1-\rho_R^{HANAPP})}. \tag{A.39}$$

**Mittlere Wartezeit einer Nachricht beim Gruppenführer:** Die Betrachtung des Senders ohne die initiale Übertragung und des Empfängers führen zu den Lastbedingungen und der mittleren Nachrichtenwartezeit des Gruppenführers:

$$\begin{aligned}\rho_G^{HANAPP} &= \lambda_r^S E(X) + (\lambda_r^S + \lambda_n^S)E(Y) + (\lambda_q^S + \lambda_{aa}^S)E(Z) \\ &\quad + \lambda_t^R E(X) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y) + \lambda_q^R E(Z+Z)\end{aligned} \tag{A.40}$$

$$\begin{aligned}E(W_G^{HANAPP}) &= \frac{\lambda_r^S E(X^2) + (\lambda_r^S + \lambda_n^S)E(Y^2) + (\lambda_q^S + \lambda_{aa}^S)E(Z^2)}{2(1-\rho_G^{HANAPP})} \\ &\quad + \frac{\lambda_t^R E(X^2) + (\lambda_{n,g}^R + \lambda_{n,r}^R)E(Y^2) + \lambda_q^R E((Z+Z)^2)}{2(1-\rho_G^{HANAPP})}.\end{aligned} \tag{A.41}$$

**Gesamtverzögerung:** Die Berechnung der Verzögerungen für die Fehlerdetektion, für die Fehlerbehebung und für die initiale Datenübertragung sind analog zur Klasse K-HNAPP. Die Durchschnittsverzögerung ist:

$$\begin{aligned}E(S_\phi^{HANAPP}) &= (1-q_D)E(I) + \Big[\sum_{i=1}^{\bar{h}-2} q_D^i (1-q_D)\Big(E(I) + E(D^{HANAPP}) + iE(R^{HANAPP})\Big)\Big] \\ &\quad + q_D^{\bar{h}-1}\Big(E(D^{HANAPP}) + (\bar{h}-1)E(R^{HANAPP})\Big).\end{aligned} \tag{A.42}$$

Für die Bestimmung der Umlaufverzögerung soll wiederum das Szenario betrachtet werden, in dem tatsächlich aggregierte Quittungen gesendet werden, obwohl dies bei K-HANAPP nur periodisch passiert. Die Umlaufverzögerung beträgt dann:

$$E(S_{RTD}^{HANAPP}) = (1-q_D)^R E(I) + \underbrace{\left(1-(1-q_D)^R\right)\left(E(D^{HANAPP})+E(R^{HANAPP})\right)}_{\ddot{U}bertragungen}$$
$$+ (h-1)\underbrace{\left[E(\widetilde{L}_{aaq}^{HANAPP})\left(T+E(W_G^{HANAPP})+E(Y)\right)\right.}_{Quittungsnachforderungen} + \underbrace{\left.E(Y)+\tau_H+E(W_G^{HANAPP})+E(Y)\right]}_{letzte\ aggregierte\ Quittung}. \quad \text{(A.43)}$$

Die Bestimmung der Maximalverzögerung $E(S_\gamma^{HANAPP})$ ist analog zur Protokollklasse K-HNAPP.

# Anhang B

# Notation der Transportprotokollanalyse

| | | |
|---|---|---|
| $B$ | - | Verzweigungsgrad des Kontrollbaums bzw. Größe einer lokalen Gruppe. |
| $B_{aa}$ | - | Anzahl an Empfängern, von denen die aggregierte Quittung beim Start der Quittungsnachforderungsphase fehlt. |
| $C_S^w, C_R^w, C_G^w$ | - | CPU-Belastung des Senders, des Empfängers, respektive des Gruppenführers. Protokollklasse $w \in \{ACK, NACK, NAPP, HACK, HNAPP, HAACK, HANAPP\}$ |
| $G$ | - | Anzahl lokaler Gruppen und Anzahl der Gruppenführer. |
| $h, \bar{h}$ | - | Maximale und durchschnittliche Höhe des Kontrollbaums. |
| $L^w$ | - | Anzahl gesendeter Quittungen pro übertragener Datennachricht. |
| $\widetilde{L}^w, \widetilde{L}_r^w$ | - | Anzahl der empfangen Quittungen beim Sender pro übertragener Datennachricht, die von allen Empfängern gesendet werden, respektive von einem beliebigen Empfänger $r$. |
| $L_a^{HAACK}$, $L_{aa}^{HAACK}$ | - | Anzahl gesendeter positiver bzw. aggregierter Quittungen. |
| $\widetilde{L}_a^{HAACK}$, $\widetilde{L}_{aa}^{HAACK}$ | - | Anzahl empfangener positiver bzw. aggregierter Quittungen. |
| $L_{aaq}^w$, $\widetilde{L}_{aaq}^w$ | - | Anzahl gesendeter bzw. empfangener Quittungsnachforderungen. |
| $L_\phi^w$ | - | Anzahl gesendeter negativer Quittungen einer Übertragungsrunde. |
| $M^w$ | - | Anzahl notwendiger Übertragungen für alle Empfänger, um eine Datennachricht erfolgreich zu empfangen. |
| $M_r^w$ | - | Anzahl notwendiger Übertragungen für einen Empfänger $r$, um eine Datennachricht erfolgreich zu empfangen. |

| | | |
|---|---|---|
| $M_u^{HAACK}$, $M_m^{HAACK}$ | - | Anzahl notwendiger Unicast- bzw. Multicast-Übertragungen, um eine Datennachricht bei allen Empfängern erfolgreich zu empfangen. |
| $M_\gamma^w$ | - | Anzahl notwendiger Übertragungen, um eine Nachricht bei allen Empfängern mit Wahrscheinlichkeit $\gamma$ erfolgreich zu empfangen. |
| $\widetilde{N}_g^w$ | - | Anzahl der Übertragungen pro Datennachricht, die von einem Gruppenführer von beliebigen Knoten empfangen wird. |
| $n_k$ | - | Anzahl Empfänger, die eine Übertragungswiederholung benötigen. |
| $N_k$ | - | Anzahl gesendeter negativer Quittungen in Runde $k$. |
| $\widetilde{N}_r^w$,$\widetilde{N}_{r,t}^w$ | - | Anzahl der Übertragungen pro Datennachricht, die von einem beliebigen Empfänger $r$ von dessen Gruppenführer empfangen werden bzw. von beliebigen Knoten auch anderer lokaler Gruppen empfangen werden. |
| $N_u$ | - | Durchschnittliche Anzahl gesendeter Unicast Nachrichten pro Übertragungswiederholung. |
| $O_r^w$, $O^w$ | - | Anzahl scheinbarer Übertragungsrunden aus der Sichtweise eines Empfänger, um eine Datennachricht an einen beliebigen Empfänger $r$ bzw. an alle Empfänger auszuliefern. |
| $O_{e,r}^w$, $O_e^w$ | - | Anzahl leerer Runden, d.h. Runden in denen aufgrund des Verlusts negativer Quittungen keine Übertragungswiederholung stattfindet, für einen beliebigen Empfänger $r$, bzw. für alle Empfänger. |
| $O_q^{HAACK}$ | - | Anzahl notwendiger Quittungsnachforderungsrunden. |
| $\hat{p}$ | - | Wahrscheinlichkeit, dass eine Quittungsnachforderung nicht erfolgreich ist. |
| $\tilde{p}$ | - | Wahrscheinlichkeit, dass eine Übertragungswiederholung notwendig ist. |
| $p_A$ | - | Wahrscheinlichkeit für den Verlust von positiven Quittungen mittels Unicast. |
| $p_{AA}$ | - | Wahrscheinlichkeit für den Verlust von aggregierten Quittungen mittels Unicast. |
| $p_c$ | - | Wahrscheinlichkeit, dass keine aggregierte Quittung gesendet werden kann aufgrund von fehlenden aggregierten Quittungen der Kindknoten. |
| $p_D$ | - | Wahrscheinlichkeit für Datenverlust bei Übertragungen mittels Unicast. |
| $p_k$ | - | Wahrscheinlichkeit, dass alle gesendeten negativen Quittungen in Runde $k$ verloren gehen. |
| $p_l$ | - | Wahrscheinlichkeit, eine Daten- oder Quittungsnachricht zu empfangen, die für eine andere lokale Gruppe bestimmt ist. |
| $p_N$ | - | Wahrscheinlichkeit für den Verlust von negativen Quittungen mittels Unicast. |
| $p_q$ | - | Wahrscheinlichkeit für den Verlust von Quittungsnachforderungen mittels Unicast. |

| | | |
|---|---|---|
| $p_s$ | - | Wahrscheinlichkeit für simultanes Senden von negativen Quittungen mittels Multicast in den Protokollklassen K-NAPP, K-HNAPP und K-HANAPP. |
| $p_t$ | - | Wahrscheinlichkeit, dass $n_k$ kleiner als der Schwellwert $\phi$ ist und folglich Unicast für die Übertragungswiederholungen verwendet wird. |
| $q_D$ | - | Wahrscheinlichkeit für Datenverlust bei Übertragungen mittels Multicast. |
| $q_N$ | - | Wahrscheinlichkeit für den Verlust von negativen Quittungen mittels Multicast. |
| $R$ | - | Anzahl der Empfänger einer Multicast-Gruppe ohne den Sender. |
| $R_N$ | - | Anzahl erfolgloser Empfänger die eine negative Quittung senden wollen. |
| $S$ | - | Anzahl periodischer Quittungen, die ein Gruppenführer empfängt. |
| $S^w_{RTD}$ | - | Mittlere Umlaufverzögerung zwischen dem initialen Eintreffen oder der Generierung einer Datennachricht beim Sender und dem erfolgreichen Empfang aller Quittungen. |
| $S^w_\phi, S^w_\gamma$ | - | Mittlere Auslieferungszeit für Datennachrichten zwischen dem initialen Eintreffen oder der Generierung beim Sender und dem erfolgreichen Empfang bei einem beliebigen Empfänger bzw. bei allen Empfängern mit der Wahrscheinlichkeit $\gamma$. |
| $T_S, T_R$ | - | Zeitgeberintervall beim Sender respektive Empfänger. |
| $W_d$ | - | Bandbreite für eine Datennachricht. |
| $W_a, W_n, W_\phi$ | - | Bandbreite für eine positive Quittung, eine negative Quittung oder eine periodische Quittung. |
| $W_{aa}, W_{aaq}$ | - | Bandbreitenbedarf für eine aggregierte Quittung bzw. eine Quittungsnachforderung. |
| $W_{aa,\phi}, W_{aaq,\phi}$ | - | Bandbreitenbedarf für eine periodische aggregierte Quittung bzw. periodische Quittungsnachforderung. |
| $W_{d,u}, W_{d,m}$ | - | Bandbreitenbedarf, um eine Datennachricht mittels Unicast bzw. Multicast zu senden. |
| $W^w_S, W^w_R, W^w_G, W^w$ | - | Bandbreitenbedarf für das Protokoll *w* des Senders, des Empfängers oder des Gruppenführers und der gesamte Bandbreitenbedarf. |
| $W^w_S, W^w_R, W^w_G$ | - | Durchschnittliche Wartezeit für eine Nachricht bis zur Bearbeitung beim Sender, Empfänger oder Gruppenführer. |
| $X$ | - | Bearbeitungszeit für Datennachrichten. |
| $X_a$ | - | Aufwand für das Empfangen und Bearbeiten einer positiven Quittung. |
| $X_{aa}$ | - | Aufwand um eine aggregierte Quittung beim Sender zu empfangen und zu bearbeiten. |
| $X_{aaq}$ | - | Aufwand um eine Quittungsnachforderung beim Sender zu bearbeiten und zu senden. |

| | | |
|---|---|---|
| $X_d$ | - | Aufwand für das Senden einer Datennachricht. |
| $X_{d,u}, X_{d,m}$ | - | Aufwand um eine Datennachricht mittels Unicast bzw. Multicast zu bearbeiten und zu senden. |
| $X_f$ | - | Aufwand um eine Nachricht von der höheren Protokollschicht zu erhalten. |
| $X_n, X_\Phi$ | - | Aufwand für das Empfangen und Bearbeiten einer negativen Quittung oder periodischen positiven Quittung. |
| $X_t$ | - | Aufwand für das Bearbeiten einer Zeitgeberunterbrechung beim Sender. |
| $Y$ | - | Bearbeitungszeit für Quittungen. |
| $Y_a$ | - | Aufwand für das Senden einer positiven Quittung beim Empfänger. |
| $Y_{aa}$ | - | Aufwand um eine aggregierte Quittung beim Empfänger zu senden. |
| $Y_{aaq}$ | - | Aufwand um eine Quittungsnachforderung beim Empfänger zu empfangen und zu bearbeiten. |
| $Y_d$ | - | Aufwand für das Empfangen und Bearbeiten einer Datennachricht. |
| $Y_f$ | - | Aufwand um eine Nachricht an die höhere Protokollschicht zu liefern. |
| $Y_n, Y_\Phi$ | - | Aufwand für das Senden einer negativen Quittung oder periodischen positiven Quittung. |
| $Y_t$ | - | Aufwand für das Bearbeiten einer Zeitgeberunterbrechung beim Empfänger. |
| $Z$ | - | Bearbeitungszeit für periodische Quittungen. |
| $\Lambda_S^w, \Lambda_R^w, \Lambda_G^w, \Lambda^w$ | - | Durchsatz bzw. Bandbreiteneffizienz des Senders, Empfängers, Gruppenführers und Gesamtdurchsatz des Protokolls $w$. |
| $\lambda$ | - | Senderate für Datennachrichten. |
| $\lambda_t^S$ | - | Initialer Datennachrichtenfluss des Senders ohne Übertragungswiederholungen. |
| $\lambda_a^S, \lambda_n^S$ | - | Nachrichtenfluss der positiven respektive negativen Quittungen, die der Sender empfängt. |
| $\lambda_r^S$ | - | Nachrichtenfluss der Übertragungswiederholungen des Senders. |
| $\lambda_s^S$ | - | Nachrichtenfluss der periodischen Quittungen, die der Sender empfängt. |
| $\lambda_t^R$ | - | Nachrichtenfluss der empfangenen Datennachrichten des Gruppenführers bei einem Empfänger. |
| $\lambda_G^R$ | - | Nachrichtenfluss der empfangenen Übertragungswiederholungen fremder lokaler Gruppen bei einem Empfänger. |
| $\lambda_n^R$ | - | Nachrichtenfluss der gesendeten negativen Quittungen eines Empfängers. |
| $\lambda_{n,g}^R$ | - | Nachrichtenfluss der mittels Multicast gesendeten negativen Quittungen eines Empfänger. |
| $\lambda_{n,r}^R$ | - | Nachrichtenfluss der empfangenen negativen Quittungen bei einem Empfänger, die mittels Multicast gesendet wurden. |

| | | |
|---|---|---|
| $\lambda_S^R$ | - | Nachrichtenfluss der periodischen Quittungen, die der Empfänger sendet. |
| $\rho_S^w, \rho_R^w, \rho_G^w$ | - | Gesamtlast des Senders, Empfängers oder Gruppenführers. |
| $\tau, \tau_H$ | - | Nachrichtenverzögerung im Netzwerk bzw. Nachrichtenverzögerung im Netzwerk bei einer hierarchischen Übertragungswiederholung. |
| $\phi$ | - | Schwellwert für Unicast– oder Multicast-Übertragungswiederholungen. Ist $n_k$ kleiner als $\phi$ wird Unicast verwendet, ansonsten Multicast. |

# Anhang C

# Glossar und Abkürzungsverzeichnis

**AACK**
Siehe aggregierte Quittung[↑].

**ACK**
Siehe positive Quittung[↑].

**Aggregierte Quittung**
Bestätigung eines Empfängers an den Vaterknoten[↑] im Kontrollbaum[↑] über den korrekten Empfang einer Datennachricht für den gesamten Teilbaum[↑].

**Anycast**
Die Übertragung einer Nachricht an einen beliebigen Empfänger aus einer Gruppe von möglichen Empfängern.

**Auslieferungsverzögerung**
Die Zeit zwischen dem initialen Eintreffen oder der Generierung einer Nachricht beim Sender und dem korrekten Empfang bei einem Empfänger.

**Blattknoten**
Hierarchische zuverlässige Multicast-Protokolle ordnen alle Empfänger in einen Kontrollbaum[↑] ein. Ein Blattknoten ist ein Knoten im Kontrollbaum der keine Kindknoten[↑] besitzt.

**Broadcast**
Die Übertragung einer Nachricht an alle Rechner in einem Netzwerk.

**CBT – Core Based Trees**
Multicast-Routing-Protokoll das ähnlich arbeitet wie PIM-SM[↑] und deshalb für verstreute Gruppen optimiert ist.

**Domäne**
Das gesamte Netzwerk wird für den Einsatz des Token-Repository-Service↑ in disjunkte Bereiche aufgeteilt, die Domänen genannt werden. Für jede Domäne gibt es einen verantwortlichen RepServer↑, den sogenannten lokalen RepServer.

**DVMRP – Distance Vector Multicast Routing Protocol**
Multicast-Routing-Protokoll das auf RPF↑ und Zurückschneiden des Spannbaums basiert.

**Empfängerinitiierte Protokolle**
Bezeichnung für Protokolle bei denen die Empfänger für die Fehlerentdeckung verantwortlich sind (siehe auch senderinitiierte Protokolle↑).

**ERS – Expanding Ring Search**
Methode um nach Vaterknoten↑ im Kontrollbaum↑ zu suchen. Dabei wird die Suche mittels Multicast an alle Kontrollbaummitglieder durchgeführt und der Ausbreitungsbereich sukzessive erhöht.

**Globale Gruppe**
Die globale Gruppe umfasst alle Mitglieder einer Multicast-Gruppe (siehe auch lokale Gruppe↑).

**GM – Globale Metrik**
Eintrag des Gruppenregisters↑ für TRS-M↑, um die untere Schranke aller Markenmetrikbeträge↑ im Gruppenbaum↑ zu kennzeichnen.

**Gruppe**
Gesamtheit aller Empfänger, die sich zum Empfang von Nachrichten an dieselbe Multicast-Adresse angemeldet haben (siehe auch globale Gruppe↑ und lokale Gruppe↑).

**Gruppenbaum**
Gruppenspezifischer Teilbaum↑ des TRS-Baums↑ für TRS-K↑ und TRS-M↑, der alle Blatt-RepServer↑ enthält, die eine Marke↑ dieser Gruppe gespeichert haben sowie alle transitiven Vaterknoten↑.

**Gruppenführer**
Ein Knoten im Kontrollbaum↑ der dafür zuständig ist, die Quittungen↑ seiner Kindknoten↑ zu sammeln und gegebenenfalls Übertragungswiederholungen zu initiieren (siehe auch lokale Gruppe↑).

**Gruppenregister**
Datenstruktur, die auf RepServern↑ für TRS-K↑ und TRS-M↑ gespeichert wird, um den Gruppenbaum↑ als Teilbaum des TRS-Baums↑ zu kennzeichnen.

**Ideale Marke**
Eine Marke↑ ist eine ideale Marke, wenn keine weiteres Marken derselben Gruppe existieren, deren Metrikbetrag kleiner ist. In TRS-M↑ kann eine erlaubte Abweichung einer idealen Marke vom global kleinsten Metrikbetrag konfiguriert werden.

**IGMP – Internet Group Management Protocol**
Protokoll zur Verwaltung von Gruppenmitgliedschaften in lokalen Netzen.

**K-ACK – Klasse ACK-basierter Multicast-Protokolle**
Protokolle dieser Klasse erfordern zwingend die Verwendung positiver Quittungen$^{\uparrow}$. Zusätzliche negative Quittungen$^{\uparrow}$ sind erlaubt.

**K-NACK – Klasse NACK-basierter Multicast-Protokolle**
Protokolle dieser Klasse verwenden nur negative Quittungen$^{\uparrow}$.

**K-NAPP – Klasse NACK-basierter Multicast-Protokolle mit NACK-suppression**
Protokolle dieser Klasse verwenden nur negative Quittungen$^{\uparrow}$ mit Quittungsunterdrückung$^{\uparrow}$.

**K-HACK – Klasse hierarchischer ACK-basierter Multicast-Protokolle**
Protokolle dieser Klasse organisieren alle Empfänger in einem Kontrollbaum$^{\uparrow}$. Es werden zur Fehlerkontrolle positive Quittungen$^{\uparrow}$ verwendet.

**K-HNAPP – Klasse hierarchischer NACK-basierter Multicast-Protokolle mit NACK-suppression**
Protokolle dieser Klasse organisieren alle Empfänger in einem Kontrollbaum$^{\uparrow}$. Es werden zur Fehlerkontrolle negative Quittungen$^{\uparrow}$ mit Quittungsunterdrückung$^{\uparrow}$ verwendet.

**K-HAACK – Klasse hierarchischer aggregierter und ACK-basierter Multicast-Protokolle**
Protokolle dieser Klasse organisieren alle Empfänger in einem Kontrollbaum$^{\uparrow}$. Es werden zur Fehlerkontrolle positive Quittungen$^{\uparrow}$ und aggregierte Quittungen$^{\uparrow}$ verwendet.

**K-HANAPP – Klasse hierarchischer aggregierter und NACK-basierter Multicast-Protokolle mit NACK-suppression**
Protokolle dieser Klasse organisieren alle Empfänger in einem Kontrollbaum$^{\uparrow}$. Es werden zur Fehlerkontrolle negative Quittungen$^{\uparrow}$ mit Quittungsunterdrückung$^{\uparrow}$ und aggregierte Quittungen$^{\uparrow}$ verwendet.

**Kindknoten**
Hierarchische zuverlässige Multicast-Protokolle ordnen alle Empfänger in einem Kontrollbaum$^{\uparrow}$ an. Die Kindknoten eines beliebigen Knotens im Kontrollbaum sind die nächsten Knoten in Richtung Blätter. Ein Blattknoten$^{\uparrow}$ hat im Gegensatz zu einem Gruppenführer$^{\uparrow}$ keine Kindknoten.

**Kontrollbaum**
Hierarchische zuverlässige Multicast-Protokolle ordnen alle Empfänger in einem Kontrollbaum an. Der Sender ist die Wurzel des Kontrollbaums. Quittungen$^{\uparrow}$ werden immer zum Vaterknoten$^{\uparrow}$ im Kontrollbaum gesendet und Übertragungswiederholungen an alle Kindknoten$^{\uparrow}$.

**Kontrollbaumhöhe**
Die Wurzel im Kontrollbaum$^{\uparrow}$ besitzt die Höhe eins. Jeder andere Knoten besitzt die Höhe des Vaterknotens$^{\uparrow}$ erhöht um eins. Die Höhe des Kontrollbaums ist die maximale Höhe aller Knoten im Kontrollbaum.

**Lokale Gruppe**
Die lokale Gruppe ist nur für hierarchische zuverlässige Multicast-Protokolle definiert. Zu einer lokalen Gruppe gehören neben einem beliebigen Knoten alle seine Kindknoten[↑] im Kontrollbaum[↑] (siehe auch globale Gruppe[↑]).

**Lokalgruppenüberlappung**
Wird der Sendebereich lokaler Gruppen[↑] bei hierarchischen Transportprotokollen nicht perfekt auf die lokale Gruppe beschränkt, so können die gesendeten Nachrichten auch innerhalb anderer lokaler Gruppen empfangen werden. Die Lokalgruppenüberlappung beschreibt diesen unerwünschten Effekt.

**Marke**
Repräsentiert das Recht für einen neuen Teilnehmer, an einem spezifizierten Vaterknoten[↑] im Kontrollbaum[↑] eine Anmeldung durchzuführen. Der referenzierte Vaterknoten agiert als Gruppenführer[↑] für den neuen Teilnehmer.

**Markencontainer**
Datenstruktur um alle Marken[↑] einer Gruppe[↑] zu speichern. Der Markencontainer enthält neben der Gruppenbezeichnung eine Menge von Markenpaketen[↑].

**Markeneigentümer**
Das Mitglied des Kontrollbaums[↑], das die Erstellung einer Marke[↑] im TRS-Dienst[↑] veranlasst hat und auf den die Marke als Vaterknoten[↑] verweist. Neue Teilnehmer wählen den Markeneigentümer als Gruppenführer[↑] im Kontrollbaum[↑].

**Markenpaket**
Datenstruktur innerhalb eines Markencontainers[↑], um alle Marken[↑] mit demselben Eigentümer[↑] zu speichern.

**MBone – Multicast Backbone**
Virtuelles Netz im Internet bestehend aus den Routern und Teilnetzen, die Multicast-Funktionalität bieten.

**MOSPF – Multicast Open Shortest Path First**
Multicast-Routing-Protokoll das auf Dijkstras Shortest-Path-Algorithmus aufbaut.

**MTP – Multicast Transport Protocol**
Flaches, zuverlässiges Multicast-Protokoll der Klasse K-NACK[↑].

**Multicast**
Die Übertragung einer Nachricht an eine Gruppe von Empfängern. Im engeren Sinne ist im Gegensatz zu Multipeer[↑] nur ein Sender erlaubt.

**Multipeer**
Die Übertragung einer Nachricht an eine Gruppe von Empfängern, wobei im Gegensatz zu Multicast[↑] explizit mehrere Sender erlaubt sind.

**NACK**
Siehe negative Quittung↑.

**Negative Quittung**
Bestätigung eines Empfängers an den Sender oder Vaterknoten↑ im Kontrollbaum↑ über den Verlust oder fehlerhaften Empfang einer Datennachricht.

**PIM-SM – Protocol Independent Multicast – Sparse Mode**
Multicast-Routing-Protokoll das für verstreute Gruppen optimiert ist und einen gemeinsamen Spannbaum für alle Sender verwendet.

**PIM-DM – Protocol Independent Multicast – Dense Mode**
Multicast-Routing-Protokoll das ähnlich arbeitet wie DVMRP↑ und deshalb für dichte Gruppen optimiert ist.

**Positive Quittung**
Bestätigung eines Empfängers an den Sender oder Vaterknoten↑ im Kontrollbaum↑ über den korrekten Empfang einer Datennachricht.

**Quittungsnachforderung**
Vom Gruppenführer↑ in hierarchischen Protokollen an die Kindknoten↑ versandte Nachricht um fehlende aggregierte Quittungen↑ nachzufordern.

**Quittung**
Bestätigung für den korrekten Empfang einer Datennachricht (positive Quittung↑, ACK) bzw. Anforderung einer Übertragungswiederholung (negative Quittung↑, NACK).

**Quittungsunterdrückung**
Zuverlässige Multicast-Protokolle können diese Technik verwenden, um möglichst wenige Quittungen↑ an den Sender oder Vaterknoten↑ im Kontrollbaum↑ zu senden. Dazu wird eine negative Quittung an die Multicast-Gruppe adressiert anstatt sie mittels Unicast direkt an den Sender zu schicken. Wird eine negative Quittung von einem anderen Empfänger empfangen, unterdrückt dieser eine evtl. zu sendende eigene negative Quittung.

**RepServer – Repository Server**
Server des TRS-Diensts↑. Wird unterschieden in Such-RepServer, die im TRS-Baum↑ nur für die effiziente Suche nach Marken↑ zuständig sind und Blatt-RepServer, die Marken speichern und die Schnittstelle des TRS-Diensts zu den Klienten sind. Jeder Blatt-RepServer ist für alle Knoten seiner Blattdomäne↑ verantwortlich.

**RMTP – Reliable Multicast Transport Protocol**
Hierarchisches, zuverlässiges Multicast-Protokoll der Klasse K-HACK↑.

**RMTP II – Reliable Multicast Transport Protocol II**
Hierarchisches, zuverlässiges Multicast-Protokoll der Klasse K-HAACK↑.

**RPF – Reverse Path Forwarding**
Algorithmus zum Fluten des Netzwerks.

**Senderinitiierte Protokolle**
Bezeichnung für Protokolle bei denen der Sender für die Fehlerentdeckung verantwortlich ist (siehe auch empfängerinitiierte Protokolle↑).

**SRM – Scalable Reliable Multicast**
Flaches, zuverlässiges Multicast-Protokoll der Klasse K-NAPP↑. Eine Besonderheit von SRM ist die Quittungsunterdrückung↑.

**TB – Teilbaum**
Eintrag des Gruppenregisters↑ für TRS-K↑, um Teilbäume↑ zu kennzeichnen, die Marken↑ enthalten. Damit wird der Gruppenbaum↑ aufgespannt.

**TBM – Teilbaum Metrik**
Eintrag des Gruppenregisters↑ für TRS-M↑, um Teilbäume↑ zu kennzeichnen, die Marken↑ enthalten einschließlich deren niedrigsten Metribetrag. Damit wird der Gruppenbaum↑ aufgespannt.

**Teilbaum**
Der Teilbaum umfasst einen Empfänger und alle transitiven Kindknoten↑ dieses Empfängers im Kontrollbaum↑, d.h. alle Kindknoten, die Kindknoten der Kindknoten usw.

**TMTP – Tree-based Multicast Transport Protokoll**
Hierarchisches, zuverlässiges Multicast-Protokoll der Klasse K-HNAPP↑.

**TRS – Token Repository Service**
Dienst, um Marken↑ zu speichern und an neue Teilnehmer zu übergeben, die in den Kontrollbaum↑ eintreten möchten. Die Implementierung des TRS-Diensts wird unterschieden in TRS-R, TRS-K und TRS-M. TRS-R verwendet die erweiternde Ringsuche (ERS)↑, während TRS-K und TRS-M auf eine hierarchische Organisation der RepServer↑ aufbauen, dem sogenannten TRS-Baum↑. TRS-M erlaubt zusätzlich, Metriken zur Markensuche zu verwenden, so dass eine ideale Marke↑ gefunden werden kann.

**TRS-Baum**
Hierarchische Organisation der RepServer↑, um eine effiziente Markensuche zu erlauben.

**Umlaufverzögerung**
Die Zeit zwischen dem initialen Eintreffen oder der Generierung einer Nachricht beim Sender und dem korrekten Empfang aller dazugehörigen Quittungen↑ nach der Auslieferung.

**Vaterknoten**
Hierarchische zuverlässige Multicast-Protokolle ordnen alle Empfänger in einem Kontrollbaum↑ an. Der Vaterknoten eines beliebigen Knotens im Kontrollbaum ist der nächste Knoten in Richtung Wurzel. Die Wurzel hat keinen Vaterknoten. Ein Vaterknoten agiert als Gruppenführer↑ seiner Kindknoten↑.

**Verzweigungsgrad**
Anzahl Kindknoten↑, die ein Gruppenführer↑ durchschnittlich im Kontrollbaum↑ besitzt. Der maximale Verzweigungsgrad ist entsprechend die maximale Anzahl Kindknoten↑ pro Gruppenführer↑. Vom Verzweigungsgrad ist die Kontrollbaumhöhe↑ abhängig.

**XTP – Xpress Transport Protocol**
Flaches, zuverlässiges Multicast-Protokoll der Klasse K-ACK↑.

# Literaturverzeichnis


Ahlers, E. (2001). “Gigabit-Ethernet: 13 PCI-Karten im Vergleich”. *c’t Magazin für Computertechnik*, (2):164–175. Verlag Heinz Heise, Hannover

Allman, M.; Glover, D.; Sanchez, L. (1999). “Enhancing TCP over satellite channels using standard mechanisms”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2488
URL: ftp://ftp.isi.edu/in-notes/rfc2488.txt

Armstrong, S.; Freier, A.; Marzullo, K. (1992). “Multicast transport protocol”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1301
URL: ftp://ftp.isi.edu/in-notes/rfc1301.txt

Bajaj, S.; Breslau, L.; Estrin, D.; Fall, K.; Floyd, S.; P. Haldar, M. H.; Helmy, A.; Heidemann, J.; Huang, P.; Kumar, S.; McCanne, S.; Rejaie, R.; Sharma, P.; Varadhan, K.; Xu, Y.; Yu, H.; Zappala, D. (1999). “Improving simulation for network research”. Technical Report 99-702, University of Southern California, USA

Ballardie, T. (1997a). “Core based trees (cbt) multicast routing architecture”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2201
URL: ftp://ftp.isi.edu/in-notes/rfc2201.txt

Ballardie, T. (1997b). “Core based trees (cbt version 2) multicast routing - protocol specification”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2189
URL: ftp://ftp.isi.edu/in-notes/rfc2189.txt

Bäurle, S. (1999). “Untersuchung zuverlässiger Multicast-Protokolle auf Anwendungsanforderungen und Fehlertoleranz”. Diplomarbeit Nr. 1717, Fakultät Informatik, Universität Stuttgart

Bhagwat, P.; Mishra, P. P.; Tripathi, S. K. (1994). “Effect of topology on performance of reliable multicast communication”. In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 602–609, Los Alamitos, USA. IEEE Press

Bhattacharyya, S.; Diot, C.; Giuliano, L.; Rockell, R.; Meylor, J.; Meyer, D.; Shepherd, G. (2000a). "A framework for source-specific ip multicast deployment". Internet Engineering Task Force, Network Working Group, Internet Draft <draft-bhattach-pim-ssm-00.txt>, work in progress
URL: http://www.ietf.org/internet-drafts/draft-bhattach-pim-ssm-00.txt

Bhattacharyya, S.; Diot, C.; Rockell, R.; Giuliano, L. (2000b). "Deployment of pim-so at sprint". Internet Engineering Task Force, Network Working Group, Internet Draft <draft-bhattach-diot-pimso-00.txt>, work in progress
URL: http://www.ietf.org/internet-drafts/draft-bhattach-diot-pimso-00.txt

Boggs, D. (1983). "Internet broadcasting". Technical Report CSL-83-3, XEROX Palo Alto Research Center, USA

Böser, M. (1999). "Simulation einer Gruppenverwaltungsstruktur für zuverlässigen Multicast". Diplomarbeit Nr. 1713, Fakultät Informatik, Universität Stuttgart

Braudes, R.; Zabele, S. (1993). "Requirements for multicast protocols". Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1458
URL: ftp://ftp.isi.edu/in-notes/rfc1458.txt

Bräunl, T. (1993). *Parallele Programmierung: eine Einführung*. Vieweg, Braunschweig/Wiesbaden

Cain, B.; Deering, S.; Thyagarajan, A. (1999). "Internet group management protocol, version 3". Internet Engineering Task Force, Network Working Group, Internet Draft <draft-ietf-idmr-igmp-v3-01.txt>, work in progress
URL: http://www.ietf.org/internet-drafts/draft-ietf-idmr-igmp-v3-01.txt

Cain, B.; Speakman, T.; Towsley, D. (2000). "Generic router assist (GRA) building block: Motivation and architecture". Internet Engineering Task Force, Network Working Group, Internet Draft <draft-ietf-rmt-gra-arch-01.txt>, work in progress
URL: http://www.ietf.org/internet-drafts/draft-ietf-rmt-gra-arch-01.txt

Cain, B.; Towsley, D. (2000). "Generic multicast transport service: Router support for multicast applications". In *Proceedings of Networking 2000, LNCS 1815*, Seiten 108–119, Paris, France

Calvert, K.; Doar, M.; Zegura, E. (1997). "Modeling internet topology". *IEEE Communications Magazine*, 35(6):160–163. IEEE Press

Carter, R. L.; Crovella, M. E. (1996). "Dynamic server selection using bandwidth probing in wide area networks". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 1014–1021, Los Alamitos, USA. IEEE Press

Chang, J.-M.; Maxemchuk, N. (1984). “Reliable broadcast protocols”. *ACM Transactions on Computer Systems*, 2(3):251–273. ACM Press

Cheriton, D. R.; Deering, S. E. (1985). “Host groups: A multicast extension for datagram internetworks”. In *Proceedings of the Ninth ACM/IEEE Data Communications Symposium*, Seiten 172–179, British Columbia, Canada. IEEE Press

Chiu, D. M.; Hurst, S.; Kadansky, M.; Wesley, J. (1998). “Tram: A tree-based reliable multicast protocol”. Technical Report TR-98-66, SUN Microsystems Research, USA

Clark, D. (1988). “The design philosophy of the darpa internet protocols”. In *Proceedings of ACM SIGCOMM Conference*, Seiten 106–114, Stanford, USA. ACM Press

Clark, D.; Chapin, L.; Cerf, V.; Braden, R.; Hobby, R. (1991). “Towards the future internet architecture”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1287
URL: ftp://ftp.isi.edu/in-notes/rfc1287.txt

Clark, D. D.; Tennenhouse, D. L. (1990). “Architectural considerations for a new generation of protocols”. In *Proceedings of ACM SIGCOMM Conference*, Seiten 200–208, Philadelphia, USA. ACM Press

Comer, D. (1979). “The ubiquitous b-tree”. *Computing Surveys*, 11(2):121–137. ACM Press

Costello, A. M.; McCanne, S. (1999). “Search party: Using randomcast for reliable multicast and local recovery”. In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 1256–1264, New York, USA. IEEE Press

Cristian, F. (1991). “Understanding fault-tolerant distributed systems”. *Communications of the ACM*, 34(2):56–78. ACM Press

Dalal, Y. K.; Metcalfe, R. M. (1978). “Reverse path forwarding of broadcast packets”. *Communications of the ACM*, 21(2):1040–1047. ACM Press

Day, J. D.; Zimmerman, H. (1983). “The OSI reference model”. *Proceedings of the IEEE*, 71(12):1334–1340. IEEE Press

Decasper, D.; Dittia, Z.; Parulkar, G.; Plattner, B. (1998). “Router plugins: A software architecture for next generation routers”. In *Proceedings of ACM SIGCOMM Conference*, Seiten 229–240, Vancouver, Canada. ACM Press

DeCleene, B. (1996). “Delay characteristics of generic reliable multicast protocols”. Technical Report TR-08150-3, Logicon TASC

Deering, S. (1989). “Host extensions for ip multicasting”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1112
URL: ftp://ftp.isi.edu/in-notes/rfc1112.txt

Deering, S.; Cheriton, D. (1985). “Host groups: A multicast extension to the internet protocol”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 966
URL: ftp://ftp.isi.edu/in-notes/rfc966.txt

Deering, S.; Estrin, D.; Farinacci, D.; Jacobson, V.; Helmy, A.; Wei, L. (1997). “Protocol independent multicast version 2, dense mode specification”. Internet Engineering Task Force, Network Working Group, Internet Draft <draft-ietf-idmr-pim-dm-05.txt>, work in progress
URL: http://www.ietf.org/internet-drafts/draft-ietf-idmr-pim-dm-05.tx

Dijkstra, E. (1959). “A note on two problems in connexions with graphs”. *Numerische Mathematik*, 1:269–271. Springer Verlag, Berlin

Diot, C.; Dabbous, W.; Crowcroft, J. (1997). “Multipoint communication: A survey of protocols, functions and mechanisms”. *IEEE Journal on Selected Areas in Communication, Special Issue on Group Communication*, 15(3):277–290. IEEE Press

Estrin, D.; Farinacci, D.; A. Helmy, D. T.; Deering, S.; Handley, M.; Jacobson, V.; Liu, C.; Sharma, P.; Wei, L. (1998). “Protocol independent multicast-sparse mode (pim-sm): Protocol specification”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2362
URL: ftp://ftp.isi.edu/in-notes/rfc2362.txt

Fenner, W. (1997). “Internet group management protocol, version 2”. Internet Engineering Task Force, Network Working Group, Request for Comments 2236
URL: ftp://ftp.isi.edu/in-notes/rfc2236.txt

Floyd, S.; Jacobson, V.; Liu, C.; McCanne, S.; Zhang, L. (1997). “A reliable multicast framework for light-weight sessions and application level framing”. *IEEE/ACM Transactions on Networking*, 5(6):784–803. IEEE, ACM Press

Graham-Cumming, J. (1997). “Hits and miss-es: A year watching the web”. *Comput. Netw. ISDN Syst.*, 29(8-13):1357–1365. Elsevier, Netherlands

Gramsamer, F. (2001). “Bimodal traffic for computer interconnects”. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2001)*, Seiten 583–589, Orlando, USA. SCS

Gu, L.; Garcia-Luna-Aceves, J. (1997). “New error recovery structures for reliable multicasting”. In *Proceedings of Sixth International Conference on Computer Communications and Networks (ICCCN ’97)*, Seiten 189–196, Las Vegas, USA. IEEE Press

Handley, M.; Jacobson, V. (1998). “SDP: Session description protocol”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2327
URL: ftp://ftp.isi.edu/in-notes/rfc2327.txt

Handley, M.; Perkins, C.; Whelan, E. (2000). “Session announcement protocol”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2974
URL: ftp://ftp.isi.edu/in-notes/rfc2974.txt

Handley, M.; Schulzrinne, H.; Schooler, E.; Rosenberg, J. (1999). “Sip: Session initiation protocol”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2543
URL: ftp://ftp.isi.edu/in-notes/rfc2543.txt

Hashimoto, T.; Yamamoto, M.; Ikeda, H.; Kurose, J. F. (1999). “Performance evaluation of reliable multicast communication protocols under heterogeneous transmission delay circumstances”. *Transactions on Communication*, E82-B(10):1609–1617. IEICE

Hofmann, M. (1998). *Skalierbare Multicast-Kommunikation in Weitverkehrsnetzen*. Infix Verlag, St. Augustin, Germany

Holbrook, H. W.; Singhal, S. K.; Cheriton, D. R. (1995). “Log-based receiver-reliable multicast for distributed interactive simulation”. In *Proceedings of ACM SIGCOMM Conference*, Seiten 328–341, Cambridge, USA. ACM Press

IEEE (2000). *Information processing systems - Local area networks - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*. IEEE 802.3/ISO 8802-3, 2000 Auflage

Jacobson, V. (1992). “A portable, public domain network whiteboard”. Technical report, Xerox PARC viewgraphs, USA

Jacobson, V.; Karels, M. J. (1988). “Congestion avoidance and control”. In *Proceedings of ACM SIGCOMM Conference*, Seiten 314–329, Stanford, USA. ACM Press

JTC1/SC6, I. (1994). “Draft on multi-peer taxonomy”. P 01.06.36 (ECFF)

Kadansky, M.; Chiu, D.; Whetten, B.; Levine, B.; Taskale, G.; Cain, B.; Thaler, D.; Koh, S. (2001). “Reliable multicast transport building block: Tree auto-configuration”. Internet Engineering

Task Force, RMT Working Group, Internet Draft <draft-ietf-rmt-bb-tree-config-02.txt>, work in progress
URL: http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-tree-config-02.txt

Kasera, S.; Kurose, J.; Towsley, D. (1997). "Scalable reliable multicast using multiple multicast groups". In *Proceedings of ACM SIGMETRICS*, Seiten 64–74, Seattle, USA. ACM Press

Kasera, S.; Kurose, J.; Towsley, D. (1998). "A comparison of server-based and receiver-based local recovery approaches for scalable reliable multicast". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 988–995, New York, USA. IEEE Press

Kleinrock, L. (1975). *Queueing Systems, Volume I: Theory*. Wiley Interscience, New York, USA

Kleinrock, L. (1976). *Queueing Systems, Volume II: Computer Applications*. Wiley Interscience, New York, USA

Kosiur, D. (1998). *IP Multicasting: The Complete Guide to Interactive Corporate Networks*. Wiley computer publishing, New York, USA

Lacher, M. (1998). "Analysis of error recovery techniques of reliable multicast protocols: centralized error recovery vs. distributed error recovery". Diplomarbeit, Technische Universität München, Institut für Informatik

Lehman, L.-w. H.; Garland, S. J.; Tennenhouse, D. L. (1998). "Active reliable multicast". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 581–589, San Francisco, USA. IEEE Press

Levine, B.; Garcia-Luna-Aceves, J. (1996). "A comparison of known classes of reliable multicast protocols". In *Proceedings of the IEEE International Conference on Network Protocols*, Seiten 112–121, Columbus, USA. IEEE Press

Levine, B.; Garcia-Luna-Aceves, J. (1998). "A comparison of reliable multicast protocols". *Multimedia Systems*, 6(5):334–348. Springer Verlag, Germany

Levine, B.; Lavo, D.; Garcia-Luna-Aceves, J. (1996). "The case for reliable concurrent multicasting using shared ack trees". In *Proceedings of the Fourth ACM Multimedia Conference (MULTIMEDIA'96)*, Seiten 365–376, New York, USA. ACM Press

Li, D.; Cheriton, D. (1998). "Oters (on-tree efficient recovery using subcasting): a reliable multicast protocol". In *Proceedings of Sixth International Conference on Network Protocols*, Seiten 237–245, Los Alamitos, USA. IEEE Press

Lin, J. J.; Paul, S. (1996). "RMTP: A reliable multicast transport protocol". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 1414–1424, San Francisco, USA. IEEE Press

Macedonia, M. R.; Brutzman, D. P. (1994). "MBone provides audio and video across the internet". *IEEE Computer*, 27(4):30–36. IEEE Press

Maihöfer, C. (2000a). "A bandwidth analysis of reliable multicast transport protocols". In *Proceedings of the Second International Workshop on Networked Group Communication (NGC 2000)*, Seiten 15–26, Palo Alto, USA. ACM Press

Maihöfer, C. (2000b). "Improving multicast ack tree construction with the token repository service". In *Proceedings of the 2000 IEEE ICDCS Workshop on Group Computation and Communication*, Seiten C57–C64, Taipeh, Taiwan. IEEE Press

Maihöfer, C. (2000c). "Scalable and reliable multicast ack tree construction with the token repository service". In *Proceedings of the IEEE International Conference on Networks (ICON)*, Seiten 351–358, Singapore. IEEE Press

Maihöfer, C. (2001). *The Token Repository Service: A Universal and Scalable Mechanism for Constructing Multicast Acknowledgement Trees*, In Sloane, A.; Lawrence, D., editors, *Multimedia Internet Broadcasting: Quality, Technology, and Interface*, Seiten 201–227. Springer Verlag, London, England

Maihöfer, C.; Rothermel, K. (1999a). "Building multicast acknowledgment trees". In *1. GI-Workshop Multicast - Protokolle und Anwendungen*, Seiten 129–143, Braunschweig, Germany

Maihöfer, C.; Rothermel, K. (1999b). "Constructing height-balanced multicast acknowledgment trees with the Token Repository Service". Technical Report TR 1999/15, University of Stuttgart

Maihöfer, C.; Rothermel, K. (2001a). "A bandwidth analysis of tree-based reliable multicast protocols". In *Proceedings of the 2001 International Symposium on the Convergence of IT and Communications (ITCom) — Scalability and Traffic Control in IP Networks*, Seiten 354–368, Denver, USA. Spie

Maihöfer, C.; Rothermel, K. (2001b). "A delay analysis of generic multicast transport protocols". In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME2001)*, Tokyo, Japan. IEEE Press

Maihöfer, C.; Rothermel, K. (2001c). "A delay analysis of reliable multicast protocols". In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2001)*, Seiten 1–9, Orlando, USA. SCS

Maihöfer, C.; Rothermel, K. (2001d). “A delay analysis of tree-based reliable multicast protocols”. In *Proceedings of the Tenth International Conference on Computer Communications and Networks*, Seiten 274–281, Scottsdale, USA. IEEE Press

Maihöfer, C.; Rothermel, K. (2001e). “A delay analysis of tree-based reliable multicast protocols”. Technical Report TR 2001/03, University of Stuttgart

Maihöfer, C.; Rothermel, K. (2001f). “Optimal branching factor for tree-based reliable multicast protocols”. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2001)*, Seiten 10–21, Orlando, USA. SCS

Maihöfer, C.; Rothermel, K. (2002). “Optimal branching factor for tree-based reliable multicast protocols”. *Computer Communications*, 25(11-12):1018–1027. Elsevier, Netherlands

Maihöfer, C.; Rothermel, K.; Mantei, N. (2000). “A throughput analysis of reliable multicast transport protocols”. In *Proceedings of the Ninth International Conference on Computer Communications and Networks*, Seiten 250–257, Las Vegas, USA. IEEE Press

Malkin, G. (1993). “Traceroute using an ip option”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1393
URL: ftp://ftp.isi.edu/in-notes/rfc1393.txt

Malkin, G. (1994). “Rip version 2: Carrying additional information”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1723
URL: ftp://ftp.isi.edu/in-notes/rfc1723.txt

Mankin, A.; Romanow, A.; Bradner, S.; Paxson, V. (1998). “Ietf criteria for evaluating reliable multicast transport and application protocols”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 2357
URL: ftp://ftp.isi.edu/in-notes/rfc2357.txt

Maufer, T. (1997). *Deploying IP Multicast in the Enterprise*. Prentice Hall, New York, USA

Merziger, G.; Wirth, T. (1993). *Repetitorium der Höheren Mathematik*. Binomi Verlag, Springe, Deutschland, 2. Auflage

Meyer, D. (1998). “Introduction to ip multicast”.
URL: http://www.nanog.org/mtg-9806/ppt/davemeyer/index.htm

Miller, C. K. (1998). *Multicast Networking and Applications*. Addison-Wesley, Massachusetts, USA

Mockapetris, P. (1987a). “Domain names - concepts and facilities”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1034
URL: ftp://ftp.isi.edu/in-notes/rfc1034.txt

Mockapetris, P. (1987b). “Domain names - implementation and specification”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1035
URL: ftp://ftp.isi.edu/in-notes/rfc1035.txt

Mohan, S.; Qian, J.; Rao, N. L. (1988). “Efficient point-to-point and point-to-multipoint selective-repeat arq schemes with multiple retransmissions: A throughput analysis”. In *Proceedings of ACM SIGCOMM Conference*, Seiten 49–57, Stanford, USA. ACM Press

Moy, J. (1994). “Multicast extensions to ospf”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1584
URL: ftp://ftp.isi.edu/in-notes/rfc1584.txt

Nonnenmacher, J.; Biersack, E. W.; Towsley, D. (1997). “Parity-based loss recovery for reliable multicast transmission”. In *Proceedings of ACM SIGCOMM Conference*, Seiten 289–300, Cannes, France. ACM Press

Nonnenmacher, J.; Lacher, M.; Jung, M.; Carl, G.; Biersack, E. (1998). “How bad is reliable multicast without local recovery”. In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 972–979, New York, USA. IEEE Press

Novell (1992). “Netware multiprotocol router configuring for performance”.
URL: http://risc.ua.edu/pub/network/netwire/novlib/01/perfor.txt

Obraczka, K. (1998). “Multicast transport protocols: a survey and taxonomy”. *Communications Magazine*, 36(1):94–102. IEEE Press

Papadopoulos, C.; Parulkar, G.; Varghese, G. (1998). “An error control scheme for large-scale multicast applications”. In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 1188–1196, New York, USA. IEEE Press

Partridge, C.; Mendez, T.; Milliken, W. (1993). “Host anycasting service”. Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1546
URL: ftp://ftp.isi.edu/in-notes/rfc1546.txt

Paul, S.; Sabnani, K.; Lin, J.; Bhattacharyya, S. (1997). “Reliable multicast transport protocol (RMTP)”. *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication*, 15(3):407–421. IEEE Press

Paul, S.; Sabnani, K. K.; Kristol, D. M. (1994). "Multicast transport protocols for high speed networks". In *Proceedings of International Conference on Network Protocols*, Seiten 4–14, Boston, USA. IEEE Press

Pejhan, S.; Schwartz, M.; Anastassiou, D. (1996). "Error control using retransmission schemes in multicast transport protocols for real-time media". *IEEE/ACM Transactions on Networking*, 4(3):413–427. IEEE, ACM Press

Pingali, S.; Towsley, D.; Kurose, J. F. (1994). "A comparison of sender-initiated and receiver-initiated reliable multicast protocols". In *Proceedings of the Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Seiten 221–230, New York, USA. ACM Press

Poo, G.; Goscinski, A. (1998). "Performance comparison of sender-based and receiver-based reliable multicast protocols". *Computer Communications*, 21(7):597–605. Elsevier, Netherlands

Postel, J. (1981a). "Internet control message protocol". Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 792
URL: ftp://ftp.isi.edu/in-notes/rfc792.txt

Postel, J. (1981b). "Internet protocol". Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 791
URL: ftp://ftp.isi.edu/in-notes/rfc791.txt

Ramakrishnan, S.; Dayal, V. (1998). "The pointcast network". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Seite 520, Seattle, USA. ACM Press

Ramakrishnan, S.; Jain, B. N. (1987). "A negative acknowledgement with periodic polling protocol for multicast over lans". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 502–511. IEEE Press

Rhee, I.; Joshi, S. R.; Lee, M.; Muthukrishnan, S.; Ozdemir, V. (2000). "Layered multicast recovery". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 805–813, Tel Aviv, Israel. IEEE Press

Rothermel, K.; Maihöfer, C. (1999). "A robust and efficient mechanism for constructing multicast acknowledgment trees". In *Proceedings of the Eight International Conference on Computer Communications and Networks*, Seiten 139–145, Boston, USA. IEEE Press

Rüegg, A. (1986). *Wahrscheinlichkeitsrechnung und Statistik: Eine Einführung für Ingenieure*. R. Oldenbourg Verlag, München

Schiller, J. (2000). *Mobile Communications*. Pearson Education Limited, London, England

Sedgewick, R. (1991). *Algorithmen*. Addison-Wesley, Bonn, 3. Auflage

Shiroshita, T.; Sano, T.; Takahashi, O.; Yamashita, M.; Yamanouchi, N.; Kushida, T. (1996). "Performance evaluation of reliable multicast transport protocol for large-scale delivery". In *Proceedings of Fifth International Workshop on Protocols for High-Speed Networks*, Seiten 149–164, Sophia Antipolis, France

Speakman, T.; Farinacci, D.; Lin, S.; Tweedly, A. (2000). "PGM reliable transport protocol specification". Internet Engineering Task Force, Network Working Group, Internet Draft <draft-speakman-pgm-spec-04.txt>, work in progress
URL: http://www.ietf.org/internet-drafts/draft-speakman-pgm-spec-04.txt

Stevens, W. R. (1994). *TCP/IP Illustrated Volume 1: The Protocols*. Addison-Wesley Longman, Amsterdam, Netherlands

Strayer, T. W.; Dempsey, B. J.; Weaver, A. C. (1992). *XTP – The Xpress transfer protocol*. Addison-Wesley Publishing Company

Svetz, K.; Randall, N.; Lepage, Y. (1996). *MBone: Multicasting Tomorrow's Internet*. IDG Books Worldwide, Inc.

Tanenbaum, A. S. (1997). *Computernetzwerke*. Prentice Hall, München, 3. Auflage

Theilmann, W. (2000). "Themenspezifische Informationssuche im Internet mit Hilfe mobiler Programme". Dissertation der Universität Stuttgart, Institut für Parallele und Verteile Höchstleistungsrechner (IPVR)

van Steen, M.; Hauck, F.; Tanenbaum, A. (1996). "A model for worldwide tracking of distributed objects". In *Proceedings of the 1996 Conference on Telecommunications Information Networking Architecture (TINA 96)*, Seiten 203–212, Heidelberg

Vinay, K. (1996). *MBone: Interactive Multimedia on the Internet*. New Riders Publishing, Indianapolis Indiana

VPNC (2001). "Virtual private network consortium". 127 Segre Place, Santa Cruz, CA 95060 USA
URL: http://www.vpnc.org

Waitzman, D.; Partridge, C.; Deering, S. (1988). "Distance vector multicast routing protocol". Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 1075
URL: ftp://ftp.isi.edu/in-notes/rfc1075.txt

Walrand, J.; Varaiya, P. (2000). *High-Performance Communication Networks*. Morgan Kaufmann Publishers, San Francisco, USA, 2. Auflage

Walz, J. (1999). "Implementierung einer Gruppenverwaltung für zuverlässigen Multicast". Studienarbeit Nr. 1737, Fakultät Informatik, Universität Stuttgart

Weiser, M. (1991). "The computer for the 21st century". *Scientific American*, Seiten 66–75

Weiser, M. (1993). "Some computer science issues in ubiquitous computing". *Communications of the ACM*, 36(7):74–83. ACM Press

Whetten, B.; Montgomery, T.; Kaplan, S. M. (1994). "A high performance totally ordered multicast protocol". In *Theory and Practice in Distributed Systems, International Workshop, Lecture Notes in Computer Science 938*, Seiten 33–57. Springer Verlag

Whetten, B.; Taskale, G. (2000). "An overview of the reliable multicast transport protocol II". *IEEE Network*, 14(1):37–47. IEEE Press

Whetten, B.; Vicisano, L.; Kermode, R.; Handley, M.; Floyd, S.; Luby, M. (2001). "Reliable multicast transport building blocks for one-to-many bulk-data transfer". Internet Engineering Task Force, Network Working Group, Request for Comments, RFC 3048
URL: ftp://ftp.isi.edu/in-notes/rfc3048.txt

Wittmann, R.; Zitterbart, M. (1999). *Multicast: Protokolle und Anwendungen*. dpunkt-Verlag, Heidelberg

Yajnik, M.; Kurose, J.; Towsley, D. (1996). "Packet loss correlation in the MBone multicast network". In *Proceedings of IEEE Global Internet*, Seiten 94–99, London, UK. IEEE Press

Yajnik, M.; Moon, S.; Kurose, J.; Towsley, D. (1999). "Measurement and modelling of the temporal dependence in packet loss". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 345–352, New York, USA. IEEE Press

Yamamoto, M.; Kurose, J.; Towsley, D.; Ikeda, H. (1997). "A delay analysis of sender-initiated and receiver-initiated reliable multicast protocols". In *Proceedings of IEEE INFOCOM Conference on Computer Communications*, Seiten 480–488, Los Alamitos. IEEE Press

Yavatkar, R.; Griffioen, J.; Sudan, M. (1995). "A reliable dissemination protocol for interactive collaborative applications". In *The Third ACM International Multimedia Conference and Exhibition (MULTIMEDIA '95)*, Seiten 333–344, New York, USA. ACM Press

Zegura, E.; Calvert, K.; Donahoo, M. (1997). "A quantitative comparison of graph-based models for internet topology". *IEEE/ACM Transactions on Networking*, 5(6):770–783. IEEE, ACM Press