

Universität Stuttgart

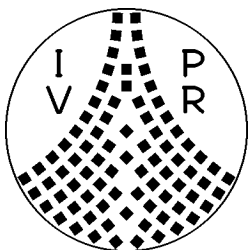
Fakultät Informatik

Prüfer: Prof. Dr. K. Rothermel
Betreuer: Dipl.-Inf. Christian Maihöfer
begonnen am 01.10.1999
beendet am 31.03.2000
CR-Klassifikation: C.2.2

Diplomarbeit Nr. 1809

Simulation von Protokollen für zuverlässige Gruppenkommunikation

Ralf Kohler



Institut für Parallele und Verteilte
Höchstleistungsrechner
Breitwiesenstraße 20-22
D-70565 Stuttgart

Danksagung

Mein herzlicher Dank gilt Christian Maihöfer, meinem Betreuer, der mir die Möglichkeit zu dieser Arbeit gegeben hat und immer Zeit für meine Fragen, für Diskussion und für aufmunternde Worte fand.

Kurzfassung

Betrachtet werden Protokolle für zuverlässige Gruppenkommunikation. Ein besonderer Schwerpunkt liegt auf den so genannten *Router-basierten* Verfahren. In Simulationen werden dann einige ausgewählte Verfahren verglichen.

Inhaltsverzeichnis

1	Einleitung	1
2	Grundlagen	3
2.1	Schichtenmodelle	4
2.1.1	Vermittlungsschicht	5
2.1.2	Transportschicht	5
2.2	Kommunikationsformen	5
2.2.1	Unicast	5
2.2.2	Broadcast	6
2.2.3	Multicast	7
2.2.4	Anycast	8
3	Multicast	9
3.1	Grundbegriffe	10
3.1.1	Offene und geschlossene Gruppen	10
3.1.2	Adressierung	10
3.1.3	Gruppenverwaltung	10
3.1.4	Nachrichtenauslieferung	10
3.1.5	Routing	11
3.2	Multicast Routing	11
3.3	Multicast Routing Protokolle	12
3.3.1	Flooding	13
3.3.2	Dense Mode Verfahren	13

3.3.3	Sparse Mode Verfahren	15
3.4	Multicast im Internet	17
3.4.1	Allgemeines	17
3.4.2	Adressierung	17
3.4.3	Reichweite von Multicastnachrichten	18
3.4.4	Der Multicast Backbone (MBone)	18
4	Zuverlässiger Multicast	21
4.1	Einführung	22
4.2	Zuverlässigkeit bei Unicast und Multicast	22
4.3	Definition der Zuverlässigkeit	22
4.4	Grundbegriffe	23
4.4.1	Sequenznummern	23
4.4.2	Pufferspeicher	23
4.4.3	Kontrollnachrichten	23
4.4.4	Flusssteuerung	23
4.5	Senderinitiierte Verfahren	24
4.6	Empfängerinitiierte Verfahren	25
4.7	Skalierbarkeitsprobleme	26
4.8	Verfahren zur Verbesserung der Skalierbarkeit	26
4.8.1	Kontrollnachrichten	26
4.8.2	Übertragungswiederholung	28
5	Nicht Router-basierte Verfahren	31
5.1	Überblick	32
5.2	Kontrollbaumaufbau	32
5.2.1	Expanding Ring Search (ERS)	32
5.2.2	Expanding Ring Advertisement (ERA)	33
5.2.3	Token Repository Service (TRS)	34
5.3	Protokolle	35
5.3.1	Log-Based Receiver Reliable Multicast (LBRM)	35
5.3.2	Scalable Reliable Multicast Protocol (SRM)	36
5.3.3	TMTP	36

6 Router-basierte Verfahren	39
6.1 Überblick über Router-basierte Ansätze	40
6.2 Lightweight Multicast Services (LMS)	40
6.2.1 Überblick	40
6.2.2 Repliers	41
6.2.3 Behandlung von NAKs	43
6.2.4 Antwort über Subcast	44
6.3 OTERS	45
6.3.1 Überblick	45
6.3.2 Das Fusion Tree Formation Protocol (FTFP)	46
6.3.3 Das Loss Recovery Protocol (LRP)	49
6.4 Active Reliable Multicast (ARM)	50
6.4.1 Überblick	50
6.4.2 Zwischenspeicherung von Multicastpaketen	51
6.4.3 NAK Vermeidung und lokale Fehlerbehebung	51
6.4.4 Isolierte Fehlerbehebung	52
6.5 Reliable Multicast Architecture (RMA)	52
6.5.1 Überblick	52
6.5.2 Addressable Internet Multicast (AIM)	53
6.5.3 Zuverlässiger Multicast mit AIM	56
6.6 Pragmatic General Multicast (PGM)	58
6.6.1 Überblick	58
6.6.2 Aufbau des Kontrollbaums	59
6.6.3 Fehlerkontrolle	60
6.6.4 Optionales Element in <i>PGM: DLRs</i>	63
6.7 Weitere Verfahren	64
6.8 Prinzipien der Router-basierten Ansätze	64

7	Vergleich ausgewählter Protokolle	67
7.1	Überblick	68
7.2	Vergleichskriterien	68
7.2.1	Aufwand Baumaufbau	68
7.2.2	Durchsatz	70
7.3	Realisierung der Simulationen	73
7.3.1	NS	73
7.3.2	Simulationsumgebung	74
7.3.3	Messung Baumaufbau	74
7.3.4	Messung Durchsatz	75
8	Simulationsergebnisse	79
8.1	Aufbau des Kontrollbaums	80
8.2	Durchsatz	82
9	Zusammenfassung und Ausblick	87
A	Projektplan Diplomarbeit	89
A.1	Einleitung	89
A.2	Beschreibung der Arbeitspakete	90
A.2.1	Definition der Arbeitspakete	90
A.2.2	Abhängigkeiten der Arbeitspakete	91
A.3	Zeitplan	91
A.4	Dokumente und Meilensteine	92
A.5	Fazit	93
B	Glossar	95
	Literaturverzeichnis	97

Kapitel 1

Einleitung

Mit der zunehmenden Verbreitung des Internet, der Vernetzung von leistungsfähigen Arbeitsplatzrechnern in den Betrieben und Privathaushalten, nimmt auch das Interesse an Anwendungsfeldern wie Konferenzsystemen, Systemen für rechnergestützte Gruppenarbeit, verteilten Netzwerkspielen oder Videoübertragungen zu. Diese Anwendungen haben die Gemeinsamkeit, dass mehrere Empfänger an den gleichen Daten interessiert sind. Für diese Anwendungen ist *Unicasting*, die weit verbreitete Kommunikationsform mit einem Sender und einem Empfänger, schlecht geeignet, da diese hier wegen der vielfach versendeten Daten zu unnötiger Netzwerkbelastung führt. Statt dessen sollte bei diesen Anwendungen *Multicasting* eingesetzt werden. Multicasting bezeichnet die effiziente Übertragung von Daten an eine Gruppe von Empfängern. Für Multicasting gibt es zahlreiche existierende Protokolle, und auch im Internet steht diese Technologie zur Verfügung. Allerdings bietet der als Erweiterung des *Internet Protocol (IP)* definierte *IP-Multicast* nur eine unzuverlässige Auslieferung der Daten. Für Video- oder Audioübertragungen kann man dies akzeptieren, da hier eine schnelle Auslieferung der Daten wichtiger ist als absolute Zuverlässigkeit. Bei rechnergestützter Gruppenarbeit, Netzwerkspielen oder auch Softwareverteilung ist aber die Zuverlässigkeit eine Voraussetzung für den Einsatz der Multicasttechnologie. Um Zuverlässigkeit zu garantieren, muss jeder Empfänger den fehlerfreien Empfang eines Datenpakets bestätigen. Sendet jeder Empfänger eine solche ACK-Nachricht¹ direkt an den Sender, so kann es durch die Vielzahl der eintreffenden ACKs bei großen Gruppen zu einer Überlastung beim Sender kommen. Um dieses in der Literatur auch als *ACK-Implosion* bezeichnete Problem zu vermeiden, kann man die Empfänger in einem hierarchisch organisierten Kontrollbaum anordnen: Jedes Gruppenmitglied sendet dann seine ACK-Nachricht an den direkten Vorgänger im Kontrollbaum. Für den Aufbau dieser Kontrollbäume sind in der Literatur verschiedene Verfahren vorgeschlagen worden. Ein weiteres Problem beim zuverlässigen Multicast sind die Übertragungswiederholungen: Ein Sender kann eine Übertragungswiederholung für fehlende oder fehlerhafte Pakete an einzelne Empfänger oder an die ganze Gruppe senden, was beides je nach Verteilung der Gruppenmitglieder und der auftretenden Fehler ineffizient

¹ von engl. Acknowledgement: Bestätigung

sein kann. Auch für diesen Aspekt der zuverlässigen Gruppenkommunikation gibt es eine Reihe von vorgeschlagenen Ansätzen.

In dieser Arbeit sollen insbesondere die so genannten *Router-basierten* Protokolle für zuverlässigen Multicast untersucht werden. Die Router-basierten Ansätze zeichnen sich dadurch aus, dass sie auf interne Funktionen der Router zugreifen und diese auch erweitern, um zum einen den Kontrollbaum an der Struktur des Multicast-Routingbaums zu orientieren und zum anderen eine möglichst optimale Übertragungswiederholung zu ermöglichen.

Nach diesem einführenden Teil wird in Kapitel 2 und Kapitel 3 auf die Grundlagen der Gruppenkommunikation eingegangen. In Kapitel 4 werden dann die Konzepte beim zuverlässigen Multicast vorgestellt. Es folgen in Kapitel 5 einige nicht Router-basierte Verfahren. In Kapitel 6 werden die Router-basierten Ansätze untersucht und einige Protokolle vorgestellt. Mittels Simulationen werden dann mehrere Protokolle verglichen. In Kapitel 7 werden die Vergleichskriterien und die Implementierung der Simulationen beschrieben. Die Simulationsergebnisse folgen in Kapitel 8. Die Zusammenfassung und ein Ausblick in Kapitel 9 schliessen die Arbeit ab.

Kapitel 2

Grundlagen

Dieses Kapitel enthält Grundlagen zur Kommunikation in Rechnernetzen, auf die eine Beschreibung der Protokolle für Gruppenkommunikation aufbaut.

Inhaltsangabe

2.1	Schichtenmodelle	4
2.2	Kommunikationsformen	5

2.1 Schichtenmodelle

Um mit der hohen Komplexität beim Entwurf und bei der Beschreibung von Netzwerken umgehen zu können, bedient man sich so genannter Schichtenmodelle. Die Zahl der Schichten oder Ebenen und deren Funktion kann von Netzwerk zu Netzwerk verschieden sein. Der Aufbau ist jedoch immer so, dass jeweils eine untere Schicht einer darüber liegenden Schicht bestimmte Dienste anbietet und die Details der Implementierung für die oberen Schichten verbirgt. Jede Schicht dieser Modelle kommuniziert (logisch) mit der entsprechenden Schicht der gleichen Ebene. Die Regel und Konventionen für diese Kommunikation bezeichnet man als *Protokoll*.

Eines der wichtigsten Modelle für die Beschreibung von Netzwerken ist das *Open Systems Interconnection (OSI) Referenzmodell* der International Standards Organization (ISO) (Abbildung 2.1.1, [Tanenbaum96]).

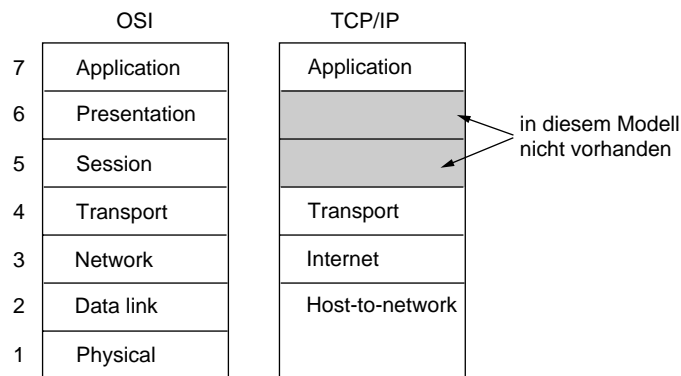


Abbildung 2.1.1: Schichtenmodelle: ISO OSI und das Internet

Die unterste Schicht des OSI-Modells (Schicht 1) ist die Bitübertragungsschicht (engl.: *physical layer*). Sie beinhaltet die physikalischen Aspekte der Datenübertragung wie zum Beispiel Kabel, Spannungen und Stecker. Darauf baut die Sicherungsschicht (Schicht 2, engl.: *data link layer*) auf. Sie ist für eine zuverlässige Datenübertragung zwischen benachbarten Stationen zuständig. Dies beinhaltet unter anderem auch die Flusssteuerung (ein schneller Sender darf einen langsamen Empfänger nicht überfluten) und eventuell eine Zugangsregelung für das Übertragungsmedium (zum Beispiel bei Funknetzen). Es folgen die Vermittlungsschicht (Schicht 3, engl.: *network layer*) und die Transportschicht (Schicht 4, engl.: *transport layer*), auf die im folgenden noch näher eingegangen werden soll. Die Kommunikationssteuerungsschicht (Schicht 5, engl.: *session layer*) beinhaltet unter anderem das Konzept, eine Arbeitssitzung über einen längeren Zeitraum (auch bei zwischenzeitlich unterbrochener Verbindung) zu unterstützen. Die Darstellungsschicht (Schicht 6, engl.: *presentation layer*) ist für eine vom Endsystem unabhängige Datendarstellung verantwortlich. Die Anwendungsschicht (Schicht 7, engl.: *application layer*) schließlich umfaßt, wie der Name schon sagt, anwendungsbezogene Dienste wie zum Beispiel elektronische Post, Dateitransfer oder World Wide Web (WWW).

Wie Abbildung 2.1.1 zeigt, fehlen im TCP/IP Modell des Internet die Kommunikationssteuerungsschicht und die Darstellungsschicht.

Auf die im Rahmen dieser Arbeit wichtigen Schichten 3 und 4 wird in den folgenden Abschnitten etwas näher eingegangen.

2.1.1 Vermittlungsschicht

Die Vermittlungsschicht (Schicht 3 nach dem OSI-Modell) wird auch als Netzwerkschicht oder Internetschicht (im TCP/IP Modell) bezeichnet. Sie ist verantwortlich für das Senden von Datenpaketen von Endsystem zu Endsystem. Die wichtigste Funktion hierbei ist die Leitwegbestimmung (engl.: *routing*). Protokolle der Vermittlungsschicht haben die Aufgabe, Datenpakete vom Ursprungsort zum Zielort zu vermitteln. Die Pakete werden auf ihrem Weg vom Senderechner zum Zielrechner in der Regel von vielen Vermittlungsrechnern (engl.: *router*) verarbeitet. Auf diesen Vermittlungsrechnern sind in der Regel nur die Schichten 1-3 des ISO-OSI Modells vorhanden. Am Zielrechner werden die Datenpakete dann an die Transportschicht weitergereicht.

Die in der Vermittlungsschicht eingesetzten Protokolle werden *Routingprotokolle* genannt. Für die Gruppenkommunikation werden spezielle Routingprotokolle eingesetzt, auf die in Kapitel 3 näher eingegangen wird.

2.1.2 Transportschicht

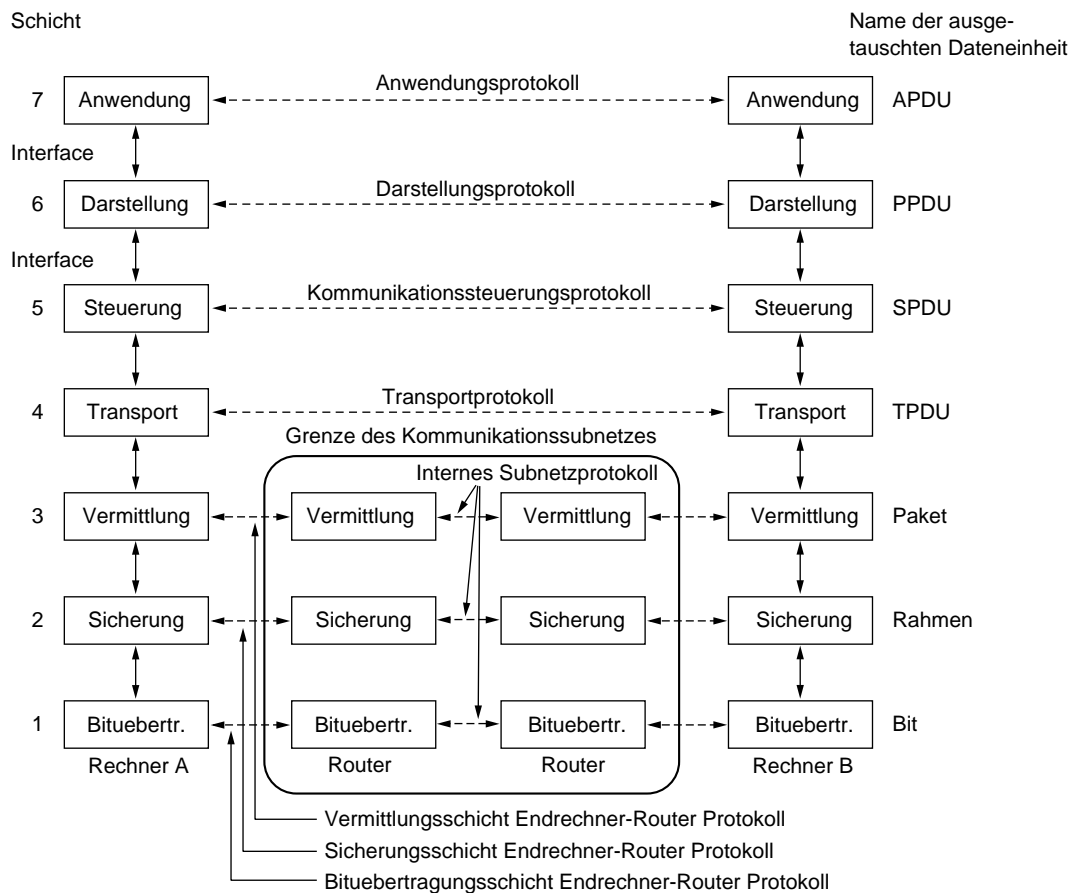
Die Transportschicht ist für die so genannte *Ende-zu-Ende-Kommunikation* zwischen Sendern und Empfängern zuständig. Hierunter versteht man auch die Möglichkeit, einzelne Prozesse auf einem Rechner adressieren zu können. In der Transportschicht werden in der Regel Funktionen implementiert, die Anwendungen vor Datenverlust, Duplizierung oder Reihenfolgeänderung im Netzwerk schützen, also einen so genannten zuverlässigen Dienst anbieten. Auch Flusssteuerung gehört in die Transportschicht.

2.2 Kommunikationsformen

Grundsätzlich lassen sich in Netzwerken verschiedene Kommunikationsformen unterscheiden. Zum besseren Verständnis sollen die für diese Arbeit wichtigsten Begriffe kurz erläutert werden.

2.2.1 Unicast

Beim *Unicast* handelt es sich um eine Kommunikation zwischen jeweils einem Sender und einem Empfänger. Abbildung 2.2.1 zeigt einen Sender, der die gleiche Nachricht jeweils an die Empfänger *A*, *B* und *C* sendet. Man sieht, dass im Netzwerk auch entsprechend drei Kopien der Nachricht transportiert werden.

Abbildung 2.1.2: Schichtenmodell: Aufbau eines *Routers*

2.2.2 Broadcast

Der Begriff *Broadcast*¹ bezeichnet die Verteilung von Informationen an *alle* erreichbaren Empfänger. Bekannt ist Broadcast beim Rundfunk und Fernsehen: Alle in der Reichweite des Senders liegenden Empfänger können das Programm empfangen, wobei der Sender keine Informationen über die Empfänger benötigt. In Netzwerken ist Broadcast in der Regel immer problemlos verfügbar, wenn gemeinsam genutzte Übertragungsmedien verwendet werden, wie zum Beispiel beim Ethernet oder in Funknetzen. Für Broadcast in anderen Netzen gibt es eine ganze Anzahl von Protokollen, wovon einige auch beim *Multicast* eingesetzt werden. Auf einige dieser Protokolle wird in 3.2 näher eingegangen. Für eine Übersicht siehe [Tanenbaum96]. Sind keine Broadcast-Funktionen vorhanden, so kann man sich unter Umständen mit einer Vielzahl von Unicast-Nachrichten behelfen (siehe Abbildung 2.2.1). Dies ist aber nur möglich, wenn der Sender alle Empfänger kennt und verschwendet natürlich Ressourcen. Ein Broadcast-Protokoll sollte idealerweise die Anzahl der über das Netzwerk verschickten Kopien der gleichen Nachricht minimieren.

¹manchmal auch als *Rundsenden* bezeichnet

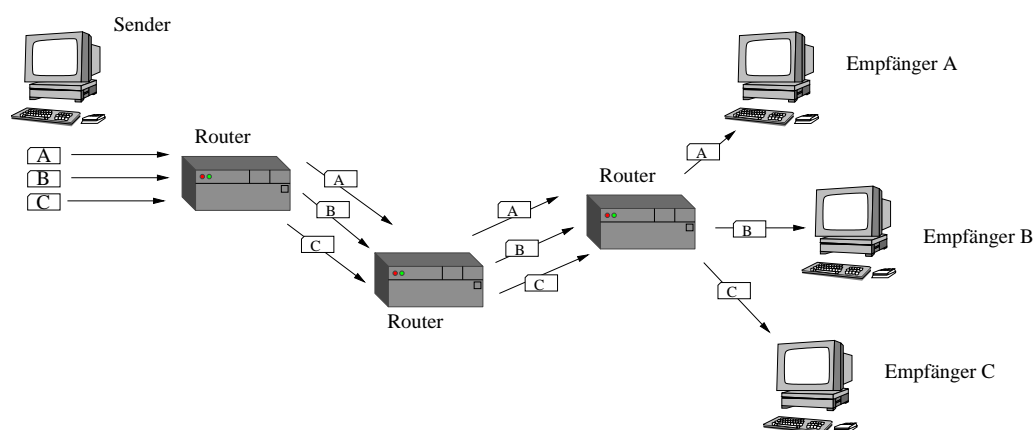


Abbildung 2.2.1: Kommunikationsformen: Unicast

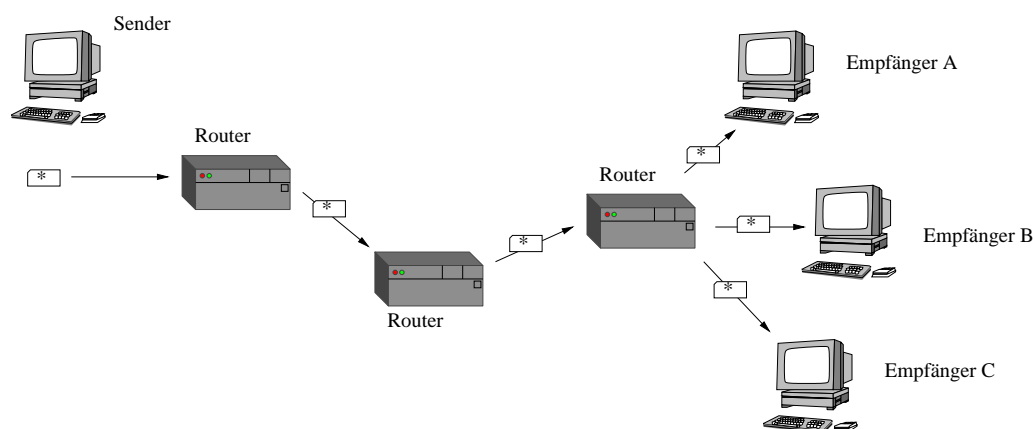


Abbildung 2.2.2: Kommunikationsformen: Broadcast

Abbildung 2.2.2 zeigt die Auslieferung einer Broadcast-Nachricht an drei Empfänger mit einem minimalem Aufwand.

2.2.3 Multicast

Gruppenkommunikation oder *Multicast*² können jetzt als Verallgemeinerung der Konzepte *Unicast* und *Broadcast* angesehen werden: Multicast bezeichnet das Senden einer Nachricht an eine definierte Gruppe von Empfängern. Unicast entspricht hiermit einem Multicast an eine Gruppe mit genau einem Empfänger, Broadcast einem Multicast an eine Gruppe mit allen möglichen Empfängern. Zusätzlich zu den Anforderungen an Broadcast-Protokolle erfordert der Multicast noch Elemente für die Gruppenverwaltung. Multicast-Protokolle sollten auch dafür sorgen, dass Nachrichten nur (einmal) über Verbindungen gesendet werden, die auch zu Gruppenmitgliedern führen. Mehr zu Multicastprotokollen in 3.2.

²Die Begriffe *Multicast* und *Gruppenkommunikation* sollen im folgenden als synonym betrachtet werden.

2.2.4 Anycast

Im Zusammenhang dieser Arbeit ist auch die Kommunikationsform des *Anycast* zu betrachten. Anycast wird im RFC 1546 [Partridge93] beschrieben, und auch die neue Version 6 des Internet Protokolls (IPv6) [Deering95] unterstützt diese Kommunikationsform. Auch beim Anycast sendet der Sender eine Nachricht an eine Gruppe. Im Gegensatz zum Multicast wird die Nachricht allerdings nur an einen Empfänger ausgeliefert. Die häufigste Anwendung dieser Kommunikationsform ist eine Nachricht an eine Gruppe von Empfängern, die den gleichen Dienst erbringen können. Die Nachricht kann dann zum Beispiel an den nächsten oder den am wenigsten belasteten Empfänger ausgeliefert werden.

Kapitel 3

Multicast

In diesem Abschnitt werden die Grundlagen der (unzuverlässigen) Gruppenkommunikation erläutert. Darauf baut dann das anschließende Kapitel über zuverlässigen Multicast auf.

Inhaltsangabe

3.1	Grundbegriffe	10
3.2	Multicast Routing	11
3.3	Multicast Routing Protokolle	12
3.4	Multicast im Internet	17

3.1 Grundbegriffe

Ein zentraler Begriff beim Multicast ist die Gruppe. Die folgenden Abschnitte beschreiben Typen von Gruppen, Fragen der Adressierung von Gruppen und der Auslieferung von Nachrichten an eine Gruppe. Für eine ausführliche Betrachtung der Grundlagen der Gruppenkommunikation siehe [Tanenbaum95].

3.1.1 Offene und geschlossene Gruppen

Eine Frage bei der Gruppenkommunikation ist, ob nur Gruppenmitglieder Nachrichten an die Gruppe senden dürfen. Ist dies der Fall, so spricht man von einer *geschlossenen Gruppe*. Dürfen auch Nichtmitglieder Nachrichten an die Gruppe senden, so spricht man von einer *offenen Gruppe*.

3.1.2 Adressierung

Jede Gruppe braucht eine eindeutige Adresse. Je nach Netzwerk gibt es verschiedene Arten von Gruppenadressen. Diese können dauerhaft zugeordnet oder bei Bedarf dynamisch vergeben werden. Als ein Beispiel wird in 3.4.2 auf Gruppenadressen im Internet eingegangen.

3.1.3 Gruppenverwaltung

Gruppenkommunikation benötigt Mechanismen für das Erzeugen und Löschen von Gruppen. Teilnehmer müssen die Möglichkeit haben, einer Gruppe beizutreten und sie wieder zu verlassen. Für die Verwaltung der Gruppenmitgliedschaft gibt es zentrale und verteilte Ansätze, wobei zentrale Ansätze beim Ausfall des zentralen Gruppenservers total versagen und schlecht skalieren. Auf das im Internet verwendete *Internet Group Management Protocol (IGMP)* wird weiter unten noch kurz eingegangen.

3.1.4 Nachrichtenauslieferung

Ein weiteres bei der Gruppenkommunikation zu lösendes Problem ist die Semantik der Nachrichtenauslieferung. Ein Aspekt hierbei ist, dass in der Regel Teilnehmer der Gruppe beitreten und die Gruppe verlassen, während Daten gesendet werden. Die Frage hierbei ist, welche Nachrichten an später hinzugekommen Mitglieder noch ausgeliefert werden. Ein weiterer Aspekt ist die Reihenfolge der Auslieferung: Werden die Nachrichten an alle Gruppenmitglieder in der gleichen Reihenfolge ausgeliefert oder spielt die Reihenfolge keine Rolle. Zuletzt ist natürlich auch eine Frage, ob die Auslieferung von Nachrichten garantiert wird, ob es also eine zuverlässige oder unzuverlässige Kommunikation gibt. Ein Beispiel für unzuverlässige Gruppenkommunikation sind die

im Internet eingesetzten Multicastprotokolle. In dieser Arbeit liegt der Schwerpunkt auf den Protokollen für zuverlässige Auslieferung.

3.1.5 Routing

Wie in 2.1.1 beschrieben, ist eine wichtige Funktion der Vermittlungsschicht die Leitwegbestimmung (engl.: *Routing*¹). Grundsätzlich geht es hierbei um die Fragestellung, auf welchen Leitungen ein Router ankommende Daten weitersenden soll. In verteilten oder zentralisierten Ansätzen wird für jedes Ziel ein so genannter *Routingbaum* aufgebaut. Abbildung 3.1.1 zeigt für ein Netzwerk einen möglichen Baum für das Ziel B. Im Internet sind die am häufigsten eingesetzten Unicast-Routingprotokolle das

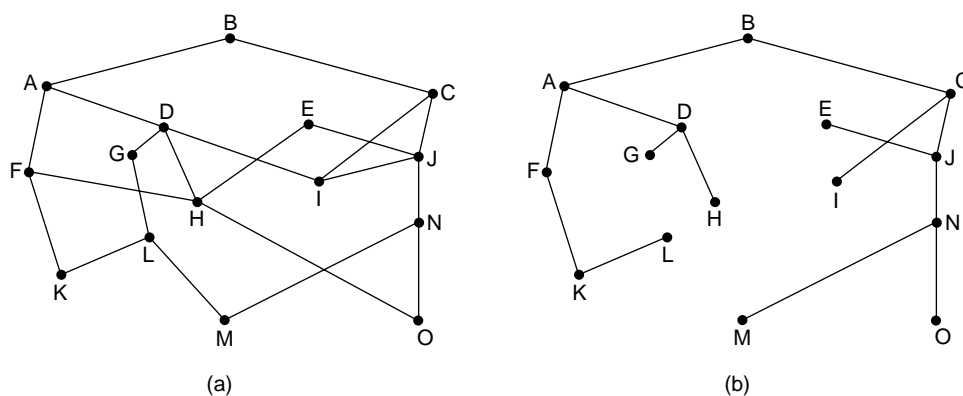


Abbildung 3.1.1: (a) Beispielnetzwerk (b) Routingbaum für Ziel B

Distance-Vector und das *Link-State* Routing. Für weitere Informationen über Unicast-Routingprotokolle siehe [Tanenbaum96].

3.2 Multicast Routing

Wie beim Unicast werden auch beim Multicast Routingbäume² aufgebaut. Abbildung 3.2.1 zeigt ein entsprechendes Beispiel. Im Beispielnetz gibt es zwei Gruppen. Abbildung (b) zeigt einen Spannbaum für alle Netzwerkknoten. Abbildungen (c) und (d) zeigen die entsprechenden Routingbäume für Gruppe 1 und Gruppe 2.

Im folgenden soll auf einige verbreitete Protokolle zum Aufbau von Multicastrooutingbäumen näher eingegangen werden.

¹Im folgenden wird nur noch der Begriff *Routing* verwendet.

²Beim Multicast auch als Spannbaum (engl: *spanning tree*) bezeichnet.

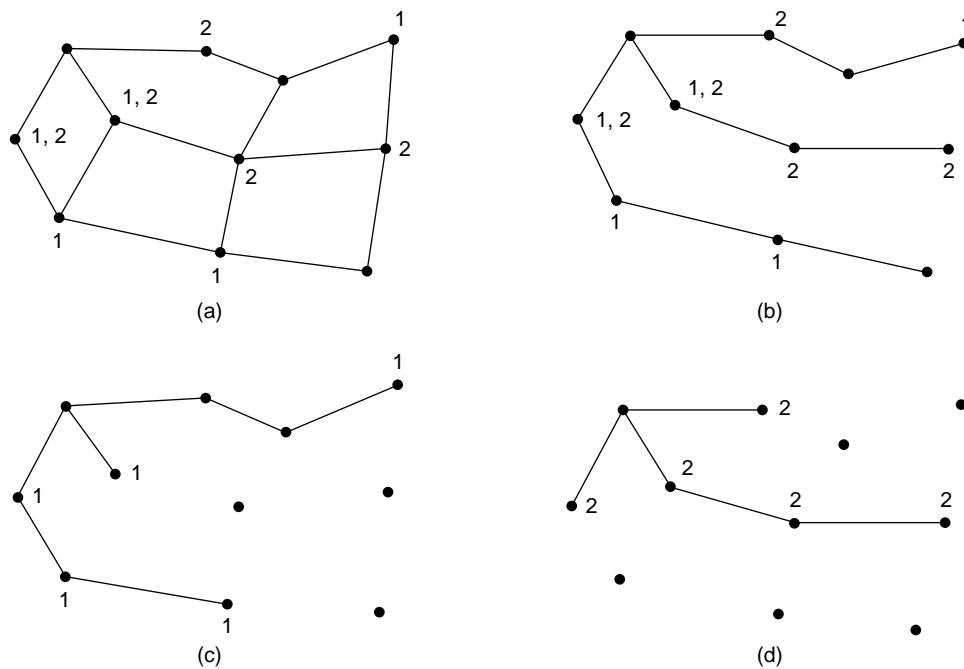


Abbildung 3.2.1: (a) ein Subnetz (b) ein Spannbaum (c) Routingbaum für Gruppe 1 (d) Routingbaum für Gruppe 2

3.3 Multicast Routing Protokolle

Wie oben erwähnt, bauen die verschiedenen Multicast Routingprotokolle einen Spannbaum auf. Dabei gibt es Protokolle, die für jede Gruppe nur einen Baum aufbauen. Diese werden *Shared Tree* Verfahren genannt. Ein ausgezeichnete Router bildet hier die Wurzel des Routingbaumes: Jeder Sender sendet seine Multicast-Daten an die Gruppe über diesen ausgezeichneten Router (oft als *Core Router* bezeichnet, siehe auch 3.3.3). Die andere Möglichkeit ist, dass für jeden Sender einer Multicastgruppe ein eigener Routingbaum aufgebaut wird. Diese Protokolle werden als *Source Routed Tree* Protokolle bezeichnet, das heißt jeder Sender bildet die Wurzel eines eigenen Multicastbaumes.

Beim Multicast werden je nach der erwarteten Verteilung der Gruppenmitglieder im Netzwerk noch zwei Ansätze bei den Routingprotokollen unterschieden: Liegen die Gruppenmitglieder dicht beieinander, so verwendet man so genannte *Dense Mode* Protokolle. Diese setzen in der Regel *Flooding* (siehe 3.3.1) ein, um Routinginformationen zu den beteiligten Routern zu verteilen.

Der andere Ansatz geht davon aus, dass Gruppenmitglieder im Netzwerk weit verteilt sind und die zur Verfügung stehende Bandbreite eher knapp bemessen ist. Verfahren für diesen Ansatz werden als *Sparse Mode* Protokolle bezeichnet. Für diese Verfahren kommen in der Regel weder *Flooding* noch *Source Routed Trees* in Frage.

3.3.1 Flooding

Flooding fällt im Zusammenhang der hier diskutierten Routingprotokolle etwas aus dem Rahmen, weil keine Spannbäume aufgebaut werden. Flooding ist ein Broadcastprotokoll. Es soll hier aber kurz erläutert werden, da es anderen Multicastprotokollen als Grundlage dient, etwa um Informationen zu allen Routern zu senden.

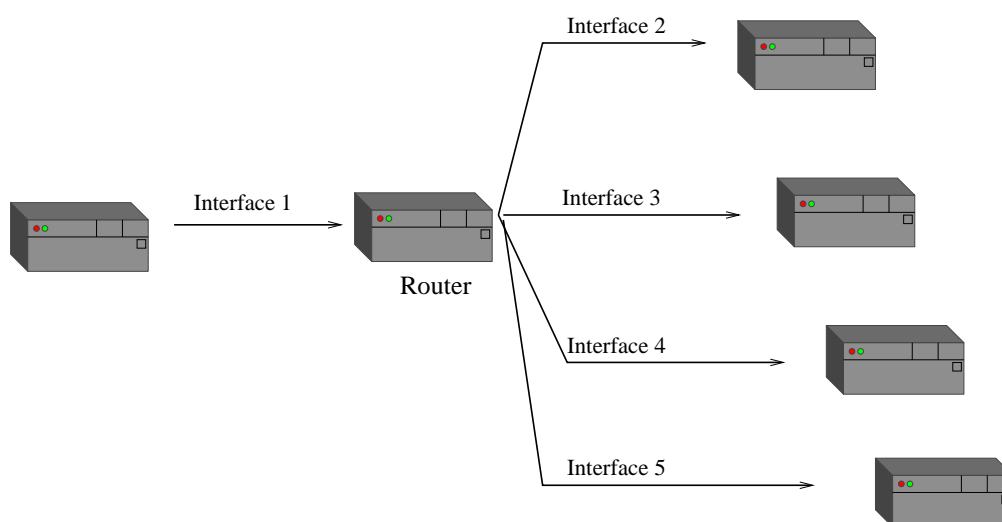


Abbildung 3.3.1: Broadcast durch Flooding

Das Grundprinzip (siehe Abbildung 3.3.1) ist ganz einfach: Ein Router sendet ein empfangenes Datenpaket auf allen Leitungen weiter, ausgenommen der, auf der das Paket empfangen wurde. Hierbei besteht natürlich die Gefahr, dass eine große Zahl von doppelten Paketen erzeugt wird und dass es zu Schleifen kommt. Dies kann durch eine Begrenzung der Lebenszeit von Paketen verhindert werden (siehe hierzu auch 3.4.3). Eine weitere mögliche Optimierung ist, dass sich der Router merkt, welche Pakete er schon gesehen hat. Pakete, die der Router zum zweiten Mal empfängt, werden verworfen.

Eine weitere Optimierung des Flooding-Algorithmus stellt das *Reverse Path Broadcasting* dar: Ein Datenpaket wird nur dann weitergeleitet, wenn es über die Netzchnittstelle empfangen wurde, über die Unicast-Pakete zum Sender geschickt würden (also die Schnittstelle, die dem kürzesten Pfad zum Sender entspricht). Dies stellt auch sicher, dass es sich bei dem empfangenen Paket nicht um ein Duplikat handelt. Um zu entscheiden, ob Pakete über diese Netzchnittstelle zum Sender geschickt würden, benötigt *RPM* ein Unicast-Routingprotokoll.

3.3.2 Dense Mode Verfahren

Wie oben erwähnt, geht man hier davon aus, dass die Gruppenmitglieder im Netzwerk „dicht“ verteilt sind, das heißt viele Subnetze enthalten mindestens ein Gruppenmitglied. In der Regel wird auch von einer hohen Bandbreite ausgegangen.

Distance Vector Multicast Routing Protocol (DVMRP)

DVMRP wird in RFC 1075 [Waitzmann88] beschrieben. Es ist ein *Source Routed Tree* Verfahren, das heißt für jeden Sender einer Gruppe wird ein eigener Multicast Baum aufgebaut (siehe oben). DVMRP setzt auf dem Unicast Distance Vector Algorithmus von Bellman-Ford [Tanenbaum96] auf. Grundsätzlich kann man bei DVMRP drei Phasen unterscheiden:

1. Beginnt ein Sender damit, Multicastpakete an eine Gruppe zu senden, so werden diese über *Reverse Path Broadcasting* (siehe 3.3.1) verteilt.
2. Erreicht die Multicastnachricht den Rand des so aufgebauten Broadcastbaumes, so senden die Router, die mit keinen Gruppenmitgliedern verbunden sind, so genannte *Prune*³ Nachrichten in Richtung Sender zurück. Durch diese werden nicht benötigte Zweige vom Baum abgeschnitten (siehe Abbildung 3.3.2).

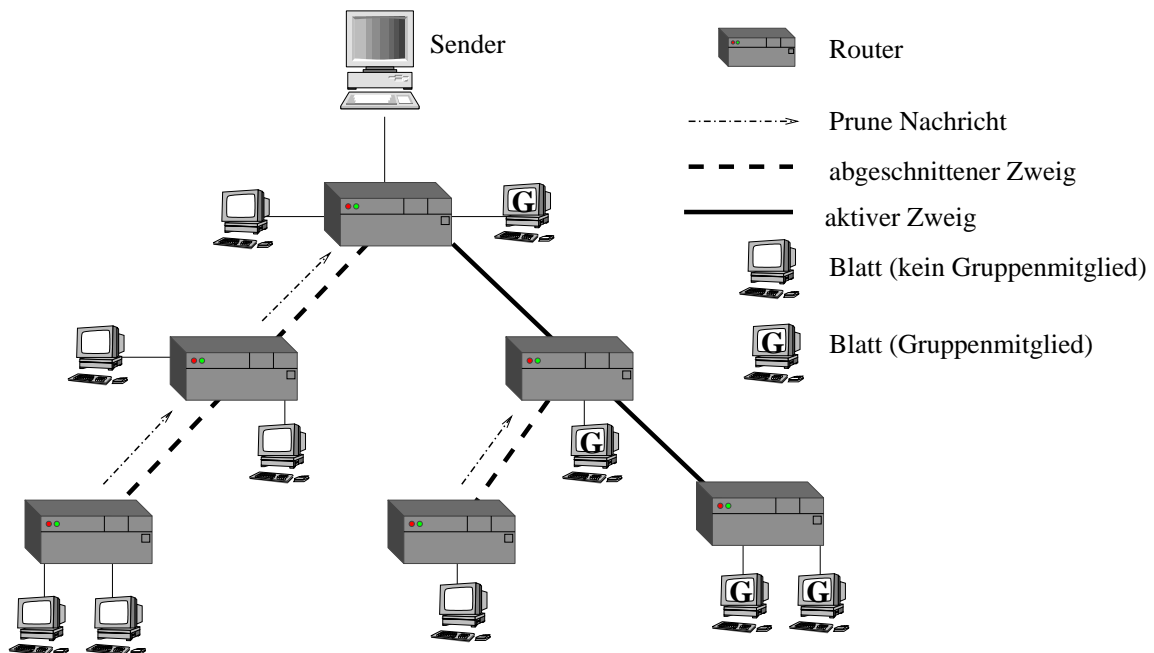


Abbildung 3.3.2: Phase 2: Zweige werden durch Prune Nachrichten abgeschnitten

3. Stellt ein Router, der zuvor eine *Prune* Nachricht geschickt hat, fest, dass es jetzt Gruppenmitglieder in seinem Subnetz gibt, versendet er so genannte *Graft*⁴ Nachrichten in Richtung Wurzel und hängt damit den abgeschnittenen Zweig wieder an.

Wie in 3.3.1 erwähnt, benötigt das *Reverse Path Broadcasting* ein Unicast-Protokoll, um zu entscheiden, ob ein ankommendes Multicast über den optimalen Pfad vom

³von engl. *to prune*: Bäume (aus-)putzen, beschneiden

⁴von engl. *to graft*: aufpfropfen

Sender empfangen wurde (und damit kein Duplikat ist). DVMRP verwendet hierzu die *Distance Vector* Unicast Routingtabelle. Als Metrik für das *Distance Vector* Verfahren wird die Anzahl der *Hops*⁵ verwendet.

Multicast Open Shortest Path First (MOSPF)

Bei MOSPF [Moy94b] handelt es sich um ein so genanntes *Link State* Protokoll. Es setzt auf dem Unicast Routingprotokoll *Open Shortest Path First (OSPF)* [Moy94a] auf. Jeder Router sammelt Informationen über seine Verbindungen zum Netzwerk und über Gruppenmitglieder. Diese Information (Link State Pakete) werden durch Flooding an alle Router verteilt. Jeder Router kennt die ganze Netzwerktopologie und kann so für jeden Sender einer Gruppe den optimalen Verteilungsbaum berechnen.

Protocol Independent Multicast - Dense Mode (PIM-DM)

Der Name *Protocol Independent Multicast (PIM)* kommt von der Tatsache, dass PIM im Gegensatz zu den beiden oben beschriebenen Verfahren kein bestimmtes Unicast Routingprotokoll voraussetzt. Ein Unicast Routingprotokoll muss aber vorhanden sein. Für PIM existiert eine *Dense Mode* (PIM-DM) und eine *Sparse Mode* (PIM-SM) Variante. Auf PIM-SM wird weiter unten noch eingegangen.

PIM-DM [Deering97] ist ziemlich ähnlich zu DVMRP. Wie DVMRP verwendet es *Reverse Path Broadcasting* mit *Prune* und *Graft* Nachrichten. Der Unterschied zu DVMRP ist, dass es mit allen Arten von Unicast Routinginformationen arbeiten kann.

3.3.3 Sparse Mode Verfahren

Die oben diskutierten Verfahren haben den Nachteil, dass sie sich nicht für weit verteilte Empfänger und Netze mit begrenzter Bandbreite eignen. Dies liegt zum einen daran, dass regelmäßig Informationen über Broadcast im ganzen Netz verteilt werden, zum anderen daran, dass durch das *Source Routed Tree* Verfahren Routinginformationen für jede $\langle \text{Sender}, \text{Gruppe} \rangle$ Kombination im Router gespeichert werden müssen. Aus diesen Gründen sind für weit verteilte Empfänger und beschränkte Bandbreiten so genannte *Sparse Mode* Verfahren vorgeschlagen worden.

Core Based Trees (CBT)

Beim *Core Based Tree (CBT)* Verfahren [Ballardie93] wird für jede Multicastgruppe nur ein Routingbaum aufgebaut. Unabhängig vom Sender werden Multicastdaten immer über diesen Baum verteilt. Der Baum enthält einen oder mehrere ausgezeichnete,

⁵ von engl. *hop*: Hopsen, Sprung. Metrik für den Abstand zweier Knoten im Netzwerk. Es werden die Anzahl der Router zwischen beiden Knoten gemessen.

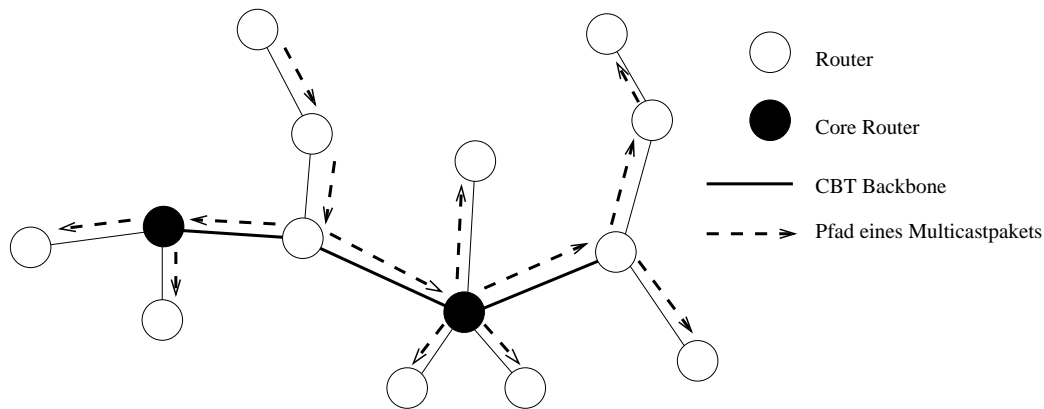


Abbildung 3.3.3: Verteilung eines Multicastpaketes über einen CBT Baum mit zwei Core Routern

so genannte *Core*⁶ *Router*. Abbildung 3.3.3 zeigt, wie Multicastpakete über einen CBT Baum verteilt werden: Die Pakete werden auf allen Netzschnittstellen des Spannbaums außer der, auf welcher das Paket eingetroffen ist, weitergeleitet.

Ein neues Gruppenmitglied hängt sich an den Baum an, indem es eine Nachricht mit dem Beitrittswunsch (eine so genannte *join*⁷ Nachricht) per Unicast an einen Core Router schickt. Alle zwischen Sender der *join* Nachricht und Core Router liegenden Router ergänzen entsprechend ihre Routingtabellen.

Bei CBT ist es auch möglich, dass Nichtmitglieder Multicastnachrichten an die Gruppe schicken: Der Sender schickt das Multicastpaket per Unicast in die Richtung eines Core Routers. Trifft das Paket auf einen Router des Multicastbaums, so wird es wie oben beschrieben über Multicast verteilt.

Probleme beim CBT Ansatz sind die Konzentration von Daten in der Nähe der Core Router und die Tatsache, dass es natürlich durch die Verwendung eines einzigen Baums für alle Sender suboptimale Pfade und größere Verzögerungen gibt.

Protocol Independent Multicast - Sparse Mode (PIM-SM)

Die zuletzt geschilderten Probleme versucht PIM-SM [Estrin97] zu umgehen, indem es die Wahl zwischen einem gemeinsamen Baum pro Gruppe (wie bei CBT) und sender-spezifischen Bäumen (wie bei den *Dense Mode* Verfahren) zuläßt. Bei der Erzeugung einer Multicastgruppe wird von PIM-SM zunächst immer ein gemeinsamer Baum wie bei CBT erzeugt. Core Router werden bei PIM-SM mit *Rendezvous Point (RP)* bezeichnet, haben aber im Grunde die gleiche Funktion. Die Besonderheit bei PIM-SM ist, dass ein Empfänger an einen Sender (nachdem er schon Daten über den gemeinsamen Baum empfängt) eine *join* Nachricht senden kann und so den Aufbau eines (optimalen) senderbasierten Baums veranlassen kann.

⁶ von engl. *core*: der Kern, das Innerste

⁷ von engl. *to join*: verbinden, sich anschließen (an)

3.4 Multicast im Internet

In diesem Abschnitt soll noch auf einige Aspekte der im Internet verwendeten Multicastkonzepte eingegangen werden, die im Zusammenhang dieser Arbeit von Interesse sind.

3.4.1 Allgemeines

Das Internet Protokoll der Version 4 (IPv4) wurde in RFC 1112 [Deering89] mit Mechanismen für Multicastkommunikation erweitert. Nicht alle IPv4 Implementierungen müssen die Erweiterung von RFC 1112 unterstützen. Als Übergangslösung wurde deshalb der *MBone* eingerichtet (siehe 3.4.4). Wie IPv4 hat auch der definierte IP-Multicast nur eine (unzuverlässige) so genannte *Best Effort* Semantik: Pakete können verloren gehen, vertauscht oder dupliziert werden. Für die Gruppenverwaltung wurde das *Internet Group Management Protocol (IGMP)* definiert. Für das Multicast Routing können grundsätzlich alle oben beschriebenen Protokolle eingesetzt werden.

3.4.2 Adressierung

Für den IP-Multicast wurden die IPv4 Adressen der Klasse D (siehe Abbildung 3.4.1) zugeordnet. Strenggenommen handelt sich hierbei nicht um Adressen, sondern um Namen für Multicastgruppen. Die neben den 4 Bit, die anzeigen, dass es sich um eine Multicastadresse handelt, noch verbleibenden 28 Bit haben im Gegensatz zu den anderen IP Adressklassen keine weitere Struktur.

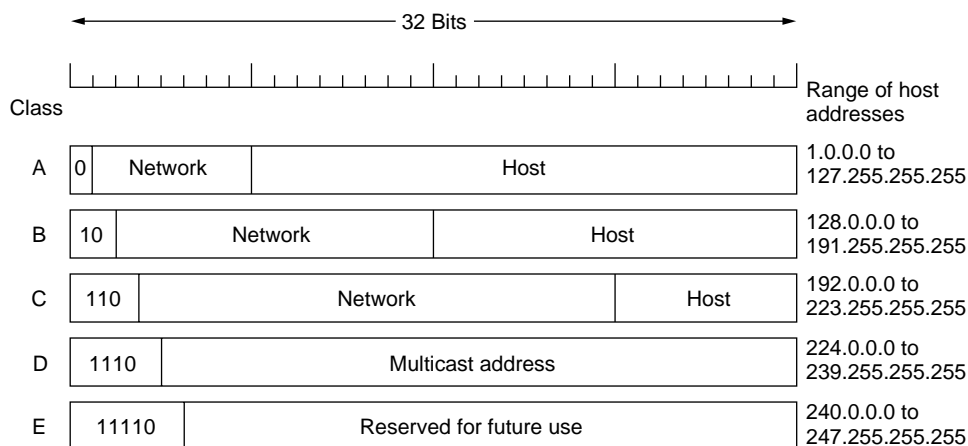


Abbildung 3.4.1: Adressformate in IPv4

3.4.3 Reichweite von Multicastnachrichten

Beim Senden von Multicastnachrichten (aber auch beim Broadcast) ist es oft von Nutzen, die Reichweite von Multicastnachrichten zu beschränken. Hierzu nutzt man eine Funktion von IP, die so genannte *Time To Live (TTL)*. Im IP-Header ist hierfür ein 8 Bit Feld vorgesehen (siehe Abbildung 3.4.2).

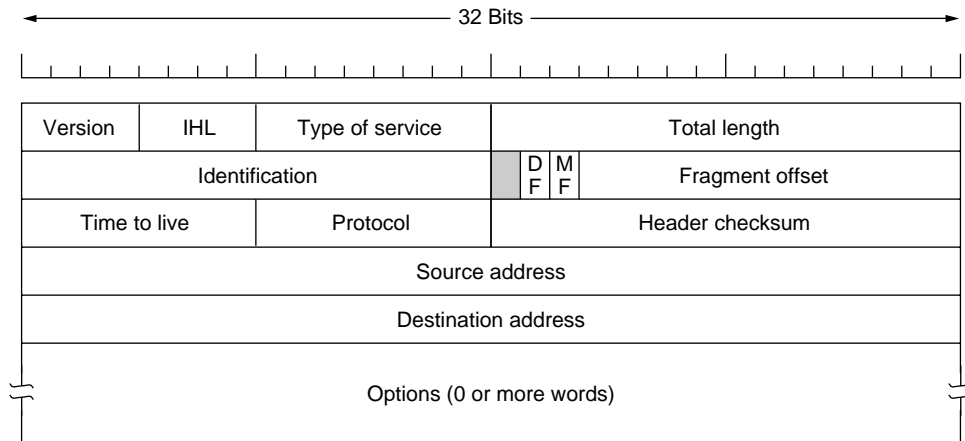


Abbildung 3.4.2: 8 bit sind im IP-Header für die *Time to live* reserviert

Auch im IP-Unicast wird die TTL benutzt, um die Reichweite von Paketen zu beschränken. Ursprünglich war der Wert im TTL-Feld als Lebenszeit des Pakets in Sekunden gedacht. Die verwendete Funktion ist dagegen ganz einfach und auch für den Multicast viel nützlicher: Der Sender setzt die TTL auf einen bestimmten Wert und jeder Router auf dem Pfad des Pakets reduziert den Wert um 1. Erreicht die TTL den Wert 0, so wird das Paket verworfen. Die TTL kann zum einen verwendet werden, um Entfernungen im Netzwerk zu messen, zum anderen ist damit ein lokal begrenzter Multicast möglich (siehe auch 5.2.1).

3.4.4 Der Multicast Backbone (MBone)

Da wie oben erwähnt die Multicasterweiterungen von RFC 1112 nicht zwingend in jeder IPv4-Implementierung vorhanden sind, ist ein Einsatz von Multicast im weltweiten Internet nicht ohne weiteres möglich, da jeder nicht multicastfähige Router entsprechende Pakete verwerfen würde. Um dennoch weltweit Multicasterweiterungen testen zu können, wurde der *Multicast Backbone (MBone)* [Kumar96] ins Leben gerufen. Der MBone besteht aus *Inseln* von multicastfähigen Routern, die durch so genannte *Tunnel* miteinander verbunden sind. Der Tunnelmechanismus funktioniert folgendermaßen: Ein IP-Multicast-Paket wird in ein IP-Unicast-Paket verpackt und vom Multicast-Router am Anfang des Tunnels zum Multicast-Router am Ende des Tunnels geschickt. Die Router dazwischen sehen nur ein Unicast-Paket. Am Ende des Tunnels wird das Multicast-Paket ausgepackt und per Multicast weitergeleitet.

Mit der flächendeckenden Einführung von IPv6, das die Multicastfähigkeit standardmäßig eingebaut hat, wird der Zwang zum Tunnelmechanismus und damit der *MBone* verschwinden.

Kapitel 4

Zuverlässiger Multicast

In diesem Kapitel soll auf die Grundlagen und Konzepte der Zuverlässigkeit bei Gruppenkommunikation eingegangen werden.

Inhaltsangabe

4.1	Einführung	22
4.2	Zuverlässigkeit bei Unicast und Multicast	22
4.3	Definition der Zuverlässigkeit	22
4.4	Grundbegriffe	23
4.5	Senderinitiierte Verfahren	24
4.6	Empfängerinitiierte Verfahren	25
4.7	Skalierbarkeitsprobleme	26
4.8	Verfahren zur Verbesserung der Skalierbarkeit	26

4.1 Einführung

Bei den bisherigen Ausführungen ist das Thema Zuverlässigkeit weitgehend ausgeklammert worden. Zum einen lag der Schwerpunkt der Betrachtungen auf der Vermittlungsschicht, zum anderen bietet der existierende IP-Multicast keine zuverlässige Übertragung an. Bisher lagen die Anwendungen im *MBone* auch hauptsächlich im Bereich der Audio- und Videokonferenzen. Bei diesen Anwendungen ist die *Best-Effort-Semantik* des IP-Multicast auch sehr geeignet: Im Audio- und Videobereich sind geringe Verzögerungen und hoher Durchsatz viel wichtiger als absolute Zuverlässigkeit.

Andere Anwendungen, die stark vom Multicast profitieren können, sind Softwareverteilung, replizierte Datenbanken, verteilte Simulationen oder auch Netzwerkspiele. Bei allen diesen Anwendungen kann aber auf Zuverlässigkeit nicht verzichtet werden.

Im weiteren Verlauf soll zunächst auf Grundlagen, Konzepte und Probleme der Zuverlässigkeit beim Multicast eingegangen werden. Nach dem in 2.1 besprochenen Schichtenmodell sind Protokolle, die der Anwendungsschicht eine zuverlässige Kommunikation ermöglichen, üblicherweise in der Transportschicht (Ebene 4) angesiedelt. Protokolle, die entsprechend aufsetzend auf dem IP-Multicast Zuverlässigkeit ermöglichen, werden dann in Kapitel 5 vorgestellt. Neue Ansätze, die Teile dieser Funktionen auch in der Vermittlungsschicht ansiedeln, werden dann in Kapitel 6 vorgestellt.

4.2 Zuverlässigkeit bei Unicast und Multicast

Zunächst stellt sich vielleicht die Frage, wieso Zuverlässigkeit beim Multicast überhaupt ein Problem darstellt. Zuverlässige Transportprotokolle für Unicastkommunikation (wie zum Beispiel das *Transmission Control Protocol (TCP)*) sind ja wohlbekannt und weit verbreitet. Ein grundsätzlicher Unterschied zwischen Unicast und den bisher eingesetzten Multicast-Verfahren ist aber die Anonymität zwischen Sender und Empfängern beim Multicast. Beim Unicast kennt der Sender trivialerweise seinen Kommunikationspartner. Beim Multicast ist das nicht so einfach: Der Sender muss nicht wissen, wer die Empfänger seiner Nachrichten sind. Wie schon in 3.4.2 erwähnt, sind die im Internet verwendeten Multicastadressen nur Bezeichner für die Gruppe und lassen keinen Schluss auf Anzahl und Ort der Empfänger zu. Ein weiteres Problem ist natürlich die potenziell sehr große Anzahl von Empfängern.

4.3 Definition der Zuverlässigkeit

Der Begriff *Zuverlässigkeit* wurde bisher nicht näher erläutert. In dieser Arbeit soll auch auf eine genauere Betrachtung der Fehlersemantik beim Multicast verzichtet werden. Zuverlässigkeit der Übertragung ist für die weiteren Betrachtungen dann gegeben, wenn alle Gruppenmitglieder alle Nachrichten empfangen. Aspekte wie der Ausfall von

Empfängern oder Netzwerkpartitionierungen sollen hier nicht betrachtet werden. Auch weiter gehende Konzepte wie die Garantie der Auslieferung innerhalb einer bestimmten Zeit (*Quality of Service*) oder *Atomizität* und *Kausalität* sollen hier nicht betrachtet werden (siehe hierfür [Tanenbaum95]).

4.4 Grundbegriffe

In diesem Abschnitt sollen noch einige Grundbegriffe erläutert werden, die im Zusammenhang mit Zuverlässigkeit verwendet werden. Grundsätzlich müssen Sender und Empfänger die Möglichkeit haben, einen Paketverlust zu entdecken und sich gegenseitig über bestimmte Ereignisse zu informieren.

4.4.1 Sequenznummern

Die Nummerierung der versendeten Pakete ist notwendig für eine zuverlässige Kommunikation. Sie ermöglicht es dem Empfänger, einen Paketverlust festzustellen, den Sender über den erfolgreichen Empfang eines Pakets zu informieren oder ein verlorenes Paket noch einmal anzufordern.

4.4.2 Pufferspeicher

Um eine zuverlässige Kommunikation zu ermöglichen, müssen Daten so lange in einem Puffer gespeichert werden, bis sichergestellt ist, dass sie alle Empfänger erhalten haben.

4.4.3 Kontrollnachrichten

Zur Anforderung von verlorenen Paketen oder zur Bestätigung des erfolgreichen Empfangs muss es die Möglichkeit zum Austausch von Kontrollnachrichten, in der Regel zwischen Empfänger und Sender, geben.

4.4.4 Flusssteuerung

Ein Protokoll zur Flusssteuerung regelt die zwischen Empfänger und Sender ausgetauschten Kontrollnachrichten, sorgt für Pufferfreigabe und verhindert einen Pufferüberlauf.

4.5 Senderinitiierte Verfahren

Der naheliegendste Ansatz für zuverlässige Gruppenkommunikation ist das so genannte *senderinitiierte Verfahren*. Hier handelt es sich um eine direkte Übertragung des beim Unicast verwendeten Verfahrens:

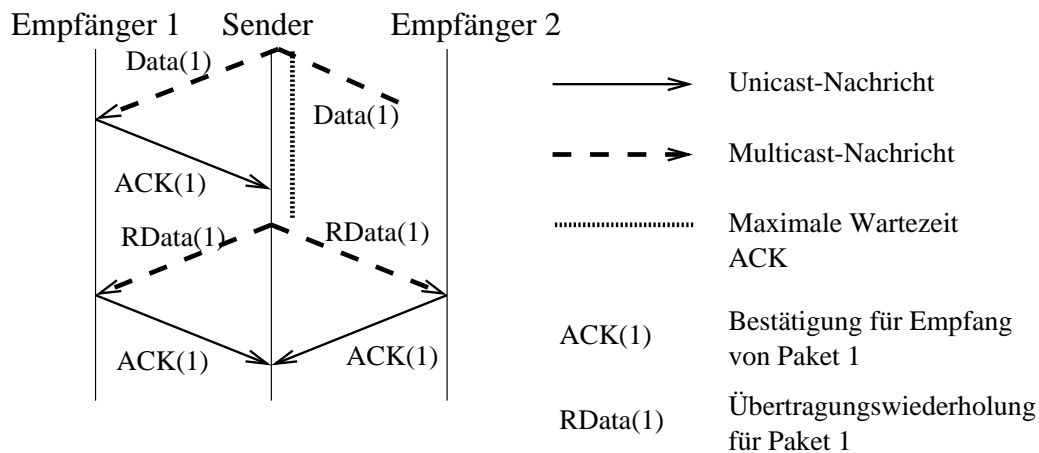


Abbildung 4.5.1: Senderinitiiertes Verfahren

Jeder Empfänger einer Multicastnachricht sendet eine Empfangsbestätigung (engl.: *Acknowledgement*, kurz *ACK*) über Unicast an den Sender zurück (siehe Abbildung 4.5.1). Der Sender kann den entsprechenden Puffer erst freigeben, wenn er von jedem Gruppenmitglied ein *ACK* bekommen hat. Sind nach einer definierten maximalen Wartezeit nicht von allen Gruppenmitgliedern *ACKs* eingetroffen, versendet der Sender die Nachricht erneut. In der Regel wird die Übertragungswiederholung per Unicast direkt an die Empfänger gesendet, die kein entsprechendes *ACK* gesendet haben. Als Optimierung kann ab einer bestimmten Anzahl von ausstehenden *ACKs* auch eine Übertragungswiederholung mittels Multicast vorgenommen werden.

Dieser Ansatz ist einfach zu implementieren, hat aber zwei schwerwiegende Probleme:

1. Der Sender muss alle Gruppenmitglieder kennen, um zu entscheiden, ob er alle *ACKs* empfangen hat: Bei sehr großen, weit verteilten Gruppen ist dieser zentrale Ansatz nicht sehr effizient (skaliert schlecht).
2. Schlimmer noch ist, dass jetzt zwar die Daten effizient über einen Spannbaum verteilt werden, der Sender aber von jedem Empfänger *ACKs* per Unicast empfängt und verarbeiten muss und bei großen Gruppen regelrecht mit *ACKs* überflutet wird. Dieses Problem wird in der Literatur allgemein als *ACK-Impllosion* bezeichnet.

4.6 Empfängerinitiierte Verfahren

Beim oben beschriebenen Verfahren lag die ganze Verantwortung für die zuverlässige Auslieferung beim Sender. Ganz anders sieht es beim empfängerbasierten Ansatz aus: Hier ist der Empfänger dafür verantwortlich, einen Paketverlust festzustellen. Durch eine Lücke in den empfangenen Sequenznummern kann er zum Beispiel erkennen, dass ein Paket verloren gegangen ist. Nach einer gewissen Wartezeit fordert er dann beim Sender eine Übertragungswiederholung an. Dies geschieht mittels einer so genannten *negativen Empfangsbestätigung* (engl.: *Negative Acknowledgement (NAK)*). Der Sender reagiert auf das NAK mit einer Übertragungswiederholung mittels Multicast (siehe Abbildung 4.6.1).

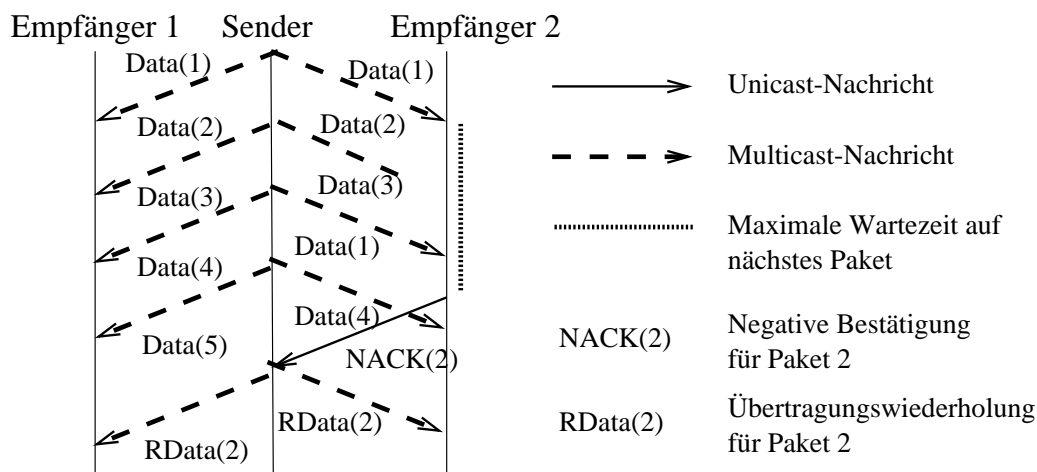


Abbildung 4.6.1: Empfängerinitiiertes Verfahren

Das Verfahren hat den großen Vorteil, dass der Sender im Gegensatz zu den empfängerbasierten Verfahren nicht alle Gruppenmitglieder kennen muss. Da der Sender nur im Fehlerfall Unicast-Kontrollnachrichten erhält, ist auch das Problem der *Implosion* beim Sender vermindert. Dennoch gibt es auch hier zwei erhebliche Nachteile:

1. Ein gravierender Nachteil des reinen empfängerinitiierten Verfahrens ist, dass der Sender keine Möglichkeit hat, festzustellen, wann er den Speicherplatz für ein bestimmtes Paket freigeben kann. Um also unter diesen Voraussetzungen absolute Zuverlässigkeit zu garantieren, müsste ein *unendlich großer Pufferspeicher* zur Verfügung stehen.
2. Ein weiteres Problem ist, dass es dennoch zu einer großen Anzahl von Unicast-NAK-Nachrichten beim Sender kommen kann: wenn es zur Netzwerküberlastung kommt, werden viele NAKs gesendet, was zu noch mehr Last, mehr Paketverlusten und noch mehr NAKs führt. Somit ist das *Implosionsproblem* auch bei diesem Ansatz nicht beseitigt.

4.7 Skalierbarkeitsprobleme

Die oben beschriebenen Ansätze weisen wie schon angedeutet größere Skalierbarkeitsprobleme auf. Diese sollen hier noch einmal zusammengefasst werden. Für eine detaillierte Betrachtung siehe [Papadopoulos98].

ACK/NAK Implosion: Dies kann zur Überlastung des Senders führen, der die Nachrichten verarbeiten muss. Außerdem kann es zur Überlastung des Netzwerks um den Sender herum führen.

Latenzzeit für Übertragungswiederholung: Hierunter versteht man die Zeit von dem Moment an, wo ein Empfänger den Verlust eines Pakets feststellt, bis zum Empfang der Übertragungswiederholung (engl.: *recovery latency*). Bei großen Netzen mit weiten Entfernungen zwischen Sendern und Empfängern kann diese Zeit bei den oben beschriebenen Verfahren sehr groß werden.

Isolation der Übertragungswiederholung: Wenn auch der Datenverlust oft nur wenige oder einzelne Empfänger betrifft, so wird die Übertragungswiederholung doch über Multicast an die ganze Gruppe gesendet, was eine unnötige Belastung des Netzwerks darstellt. Im Extremfall kann ein einziger Empfänger, der an einer stark verlustbehafteten Netzwerkverbindung hängt, die ganze Gruppe mit Übertragungswiederholungen überfluten. Dieses Phänomen wird in der Literatur auch als *crying baby problem* [Quinn98] bezeichnet. Isolation der Übertragungswiederholung (engl.: *recovery isolation*) ist ein Maß, das zeigt, wieviele Empfänger eine Übertragungswiederholung empfangen haben im Verhältnis zu der Zahl von Empfängern, die wirklich vom Datenverlust betroffen sind.

Änderung der Gruppe: Das in 4.5 beschriebene Verfahren skaliert sehr schlecht bei starken Veränderungen der Gruppenmitgliedschaft, da der Sender über alle Änderungen informiert werden muss.

4.8 Verfahren zur Verbesserung der Skalierbarkeit

Für die oben beschriebenen Probleme sollen jetzt einige Lösungsansätze vorgestellt werden. Im nächsten Kapitel werden dann einige entsprechende Protokolle vorgestellt. Im Kapitel 6 werden dann die Verfahren vorgestellt, die Änderungen in Schicht 3 erfordern.

4.8.1 Kontrollnachrichten

Die folgenden drei Ansätze haben einen besseren Umgang mit den Kontrollnachrichten und eine Lösung des Implosionsproblems zum Ziel.

Vermeidung von NAK-Nachrichten

Hier geht es um die Vermeidung von NAK-Nachrichten bei empfängerbasierten Verfahren (engl.: *Receiver Initiated with NAK Avoidance (RINA)*).

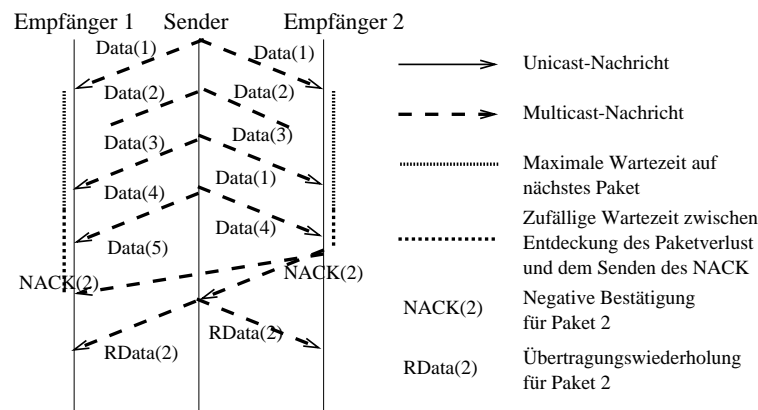


Abbildung 4.8.1: Empfängerbasierter Ansatz mit NAK-Vermeidung

Die Funktionsweise ist folgendermaßen (siehe Abbildung 4.8.1): Wenn ein Empfänger einen Paketverlust feststellt, wartet er erst eine zufällige Zeitspanne und schickt dann ein NAK über Multicast an die ganze Gruppe (also an den Sender und alle Empfänger). Erhält ein Empfänger, während er darauf wartet, ein NAK zu versenden, das entsprechende NAK von einem anderen Empfänger, so verzichtet er darauf, selbst ein NAK zu senden und verhält sich so, als habe er eines gesendet. Im optimalen Fall wird von einer Gruppe von betroffenen Empfängern nur ein NAK zum Sender geschickt.

Baumbasierte Verfahren

Die baumbasierten Verfahren sind ein hierarchischer Ansatz: Die Gruppenmitglieder werden logisch in einem so genannten *Kontrollbaum* angeordnet. Jeder Knoten im Kontrollbaum sendet Kontrollnachrichten nur an seinen direkten Vorgängerknoten. Hat ein Knoten von allen Nachfolgern ein ACK bekommen, sendet er nur *ein* akkumuliertes ACK an seinen Vorgänger weiter. Dieses Verfahren zeigt gute Skalierbarkeit und kann eine Implosion von Kontrollnachrichten beim Sender wirksam verhindern.

In der Literatur sind verschiedene Verfahren zum Aufbau des Kontrollbaums vorgeschlagen worden. Einige davon werden in Kapitel 5 vorgestellt. Die in Kapitel 6 beschriebenen Router-basierten Ansätze verwenden in der Regel den Routingbaum als Kontrollbaum.

Ringbasierte Verfahren

Ringbasierte Verfahren ordnen die Gruppenmitglieder in einem logischen Ring. Alle verwendeten Verfahren basieren auf dem in [Chang84] beschriebenen *Token Ring Protocol (TRP)*. Innerhalb der Gruppe ist immer der Inhaber des Tokens für das Senden

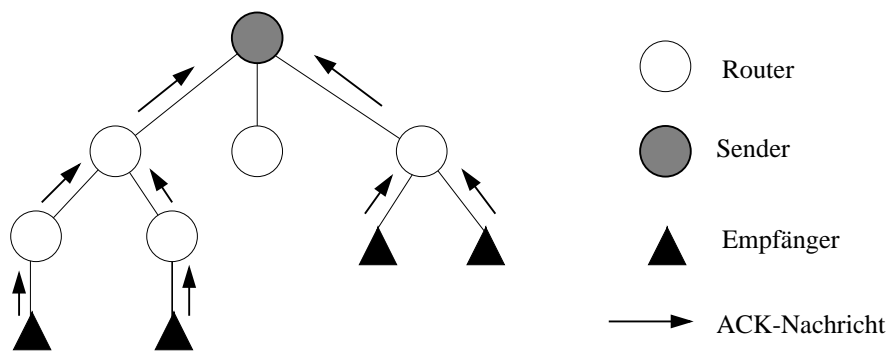


Abbildung 4.8.2: Baumbasierter Ansatz

eines ACKs verantwortlich. Empfängt er ein Multicast-Datenpaket, so versendet er ein ACK über Multicast an die ganze Gruppe (und den Sender). Dieses ACK dient auch der Tokenweitergabe: Für das nächste Datenpaket ist dann der Nachfolger im Ring zuständig. Die ACKs können auch als Zeitstempel dienen, um die Pakete beim Empfänger in einer Totalordnung auszuliefern.

Das innerhalb des Rings verwendete Verfahren ist rein empfängerinitiiert: Stellt ein Empfänger einen Datenverlust fest, so sendet er über Unicast ein NAK an den Tokenhalter, der dann wieder über Unicast eine Übertragungswiederholung sendet (siehe Abbildung 4.8.3).

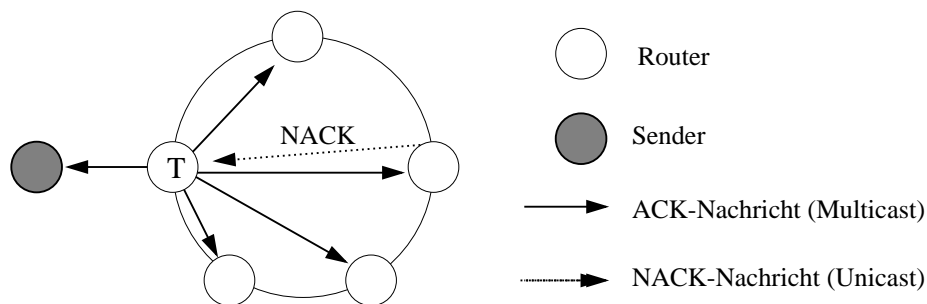


Abbildung 4.8.3: Ringbasierter Ansatz

Das Verfahren vermeidet das Implosions-Problem durch Verteilung der Aufgaben im Ring. Es hat aber die gleiche Schwäche wie andere empfängerbasierte Verfahren, dass es für absolute Zuverlässigkeit unendlich viel Speicher benötigt, da die Tokenhalter das Paket, für das sie die Verantwortung übernommen haben, nie freigeben dürfen. Ein weiterer Nachteil des Verfahrens ist der Aufwand für das Tokenmanagement.

4.8.2 Übertragungswiederholung

Bisher ging es vor allem um Ansätze, die bessere Effizienz und Skalierbarkeit im Umgang mit Kontrollnachrichten zeigen. Jetzt soll noch kurz auf Verbesserungen bei der Übertragungswiederholung eingegangen werden.

Subcast

Subcast wird in 6.2.1 näher beschrieben (siehe auch [Papadopoulos98]). Grundsätzlich wird durch Subcast eine Verbesserung der Isolationseigenschaft erreicht, indem eine Übertragungswiederholung nur an einen Unterbaum des Routingbaums gesendet wird.

Übertragungswiederholung nicht durch Sender

Eine Verbesserung der Latenzzeit, eine bessere Isolation und auch eine Entlastung des Senders kann man dadurch erreichen, dass Empfänger eine Übertragungswiederholung nicht beim Sender, sondern bei (lokalen) ausgezeichneten Empfängern¹ anfordern. Auf entsprechende Realisierungen und Konzepte wird bei der Beschreibung der in Kapitel 5 und Kapitel 6 vorgestellten Verfahren eingegangen.

¹oft engl.: *Designated Receiver (DR)*

Kapitel 5

Nicht Router-basierte Verfahren

In diesem Kapitel sollen einige nicht Router-basierte Verfahren für zuverlässige Gruppenkommunikation betrachtet werden. Der Schwerpunkt der Betrachtung liegt hierbei auf dem Aufbau des Kontrollbaums.

Inhaltsangabe

5.1	Überblick	32
5.2	Kontrollbaumaufbau	32
5.3	Protokolle	35

5.1 Überblick

In den folgenden Abschnitten werden einige Verfahren zum Aufbau eines Kontrollbaumes beschrieben. Danach werden einige existierende Protokolle für zuverlässigen Multicast vorgestellt.

5.2 Kontrollbaumaufbau

5.2.1 Expanding Ring Search (ERS)

Die *Expanding Ring Search (ERS)* ist ein verbreitetes Verfahren, um in einem Netzwerk nach bestimmten Ressourcen zu suchen. Im Zusammenhang mit dem Aufbau eines Kontrollbaumes für zuverlässigen Multicast wurde das Verfahren in [Yavatkar95] vorgestellt. Bei dem beschriebenen Verfahren wird davon ausgegangen, dass schon ein Kontrollbaum besteht, an den sich ein Knoten anhängen will. Im Extremfall besteht der Kontrollbaum nur aus der Wurzel (dem Sender). Die Initiative geht bei diesem Ansatz vom neuen Gruppenmitglied aus, das sich an den Baum anhängen will, weshalb man hier auch von einer *kindbasierten* Eingliederung oder einer außerhalb des Baumes beginnenden Eingliederung (engl. *off-tree approach*) spricht.

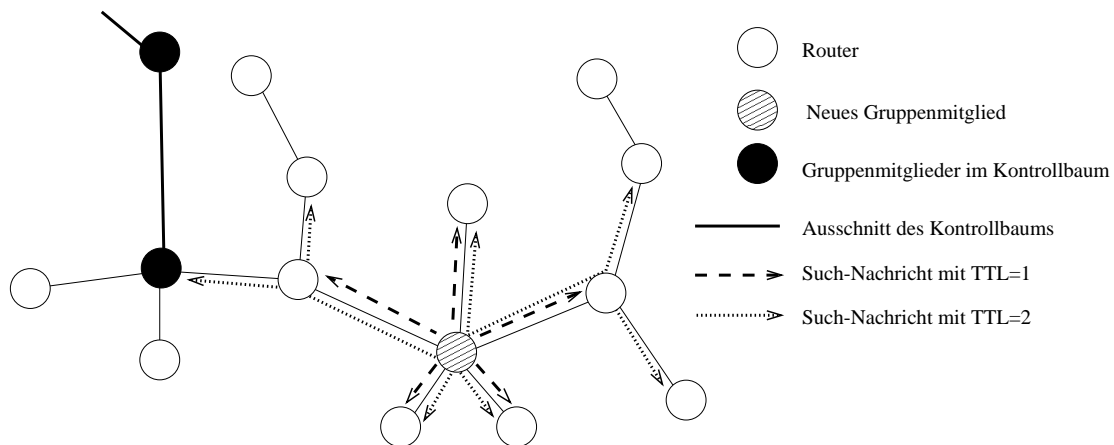


Abbildung 5.2.1: Zunehmender Suchradius mit ERS

Abbildung 5.2.1 zeigt den Ablauf: Ein neues Gruppenmitglied sendet über Multicast Suchnachrichten (in [Yavatkar95] als *SEARCH_FOR_PARENT* bezeichnet) aus. Dabei wird ein Multicast mit begrenzter Reichweite verwendet (siehe 3.4.3). Begonnen wird mit einem kleinen Wert für die TTL (im Beispiel $TTL = 1$). Die TTL wird so lange erhöht, bis ein Knoten des Kontrollbaumes erreicht wird, der bereit ist, den neuen Knoten als Nachfolgerknoten zu akzeptieren. Dieser antwortet dann per Unicast mit einer *WILLING_TO_BE_PARENT* Nachricht. Treffen mehr als eine solche Nachricht ein, so kann der neue Knoten den (basierend auf der TTL) am nächsten gelegenen Knoten wählen.

ERS sorgt dafür, dass die Entfernungen in Kontrollbaum (mit der TTL als Maß) optimiert werden. Ein Problem ist, dass dies nur bei *Source Routed Tree* Routingprotokollen (siehe 3.3) funktioniert, bei *Shared Tree* Verfahren wird auf die oben beschriebene Weise nicht unbedingt der nächste Knoten gefunden. Ein anderes Problem ist, dass die TTL nicht immer ein gutes Maß ist: Ein Knoten kann zwar weniger Hops entfernt sein, aber die Verbindung kann trotzdem eine größere Verzögerung und geringere Bandbreite haben als die zu einem Knoten, der bezüglich der Hops weiter entfernt ist.

5.2.2 Expanding Ring Advertisement (ERA)

Eine Alternative zu dem oben beschriebenen Ansatz ist, dass die Initiative von den Knoten des schon bestehenden Kontrollbaums ausgeht. Man spricht dann von einer *vaterbasierten* Eingliederung oder von einer innerhalb des Baumes beginnenden Eingliederung (*on-tree approach*). Hier versenden Knoten des Kontrollbaumes, die bereit sind, Nachfolgerknoten zu akzeptieren, periodisch so genannte Einladungsnachrichten. Diese werden wie bei ERS mit begrenztem Multicast verschickt (siehe Abbildung 5.2.2).

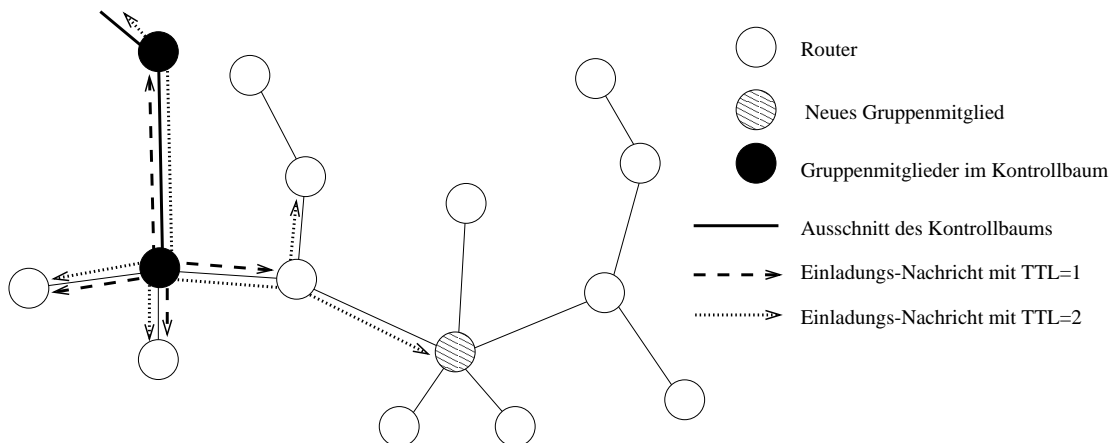


Abbildung 5.2.2: Einladungsnachrichten bei ERA

Dieses Verfahren ist in [Lin96] mit einem begrenzten Multicast und einer vorgegebenen festen TTL beschrieben. Um die Belastung für das Netzwerk zu minimieren und zu erreichen, dass möglichst lokale Knoten eine Einladung bekommen, können die Nachrichten aber auch mit einer zunehmenden TTL verschickt werden [Levine96]. Zusätzlich kann die Frequenz der Aussendung von Einladungsnachrichten mit zunehmender TTL abnehmen. Eine weitere Variante ist, dass die Einladungsnachrichten mit einer nach einer bestimmten Verteilungsfunktion gewählten TTL ausgesendet werden [Hofmann96].

Analog zu ERS ist es auch bei ERA so, dass der neue Knoten bei mehr als einer empfangenen Einladung die mit der kleinsten Entfernung (größten TTL) wählt.

Der Ansatz hat grundsätzlich die gleichen Nachteile wie ERS. Ein zusätzlicher Nachteil ist, dass auch Einladungsnachrichten verschickt werden müssen, wenn sich gar kein

neuer Knoten an den Baum anhängen möchte. ERA kann allerdings selbst dann zum Einsatz kommen, wenn die beitretenden Knoten keine Multicastnachrichten senden können, was bei ERS nicht möglich ist.

5.2.3 Token Repository Service (TRS)

Das in [Rothermel99] beschriebene Verfahren ist wie ERS ein *Off-Tree* Ansatz: Ein neues Gruppenmitglied sucht einen Knoten im Kontrollbaum, an den es sich anhängen kann. Anstatt aber Suchnachrichten auszusenden, wendet es sich an den so genannten *Token Repository Service (TRS)*. Der TRS liefert ein Token und entfernt es aus dem Repository (siehe Abbildung 5.2.3). Im Token ist die Information enthalten, bei welchem Vorgängerknoten sich das neue Mitglied anmelden kann. Der neue Kontrollbaumknoten stellt dem TRS so viele Tokens von sich zur Verfügung, wie er selbst Nachfolgerknoten akzeptieren möchte.

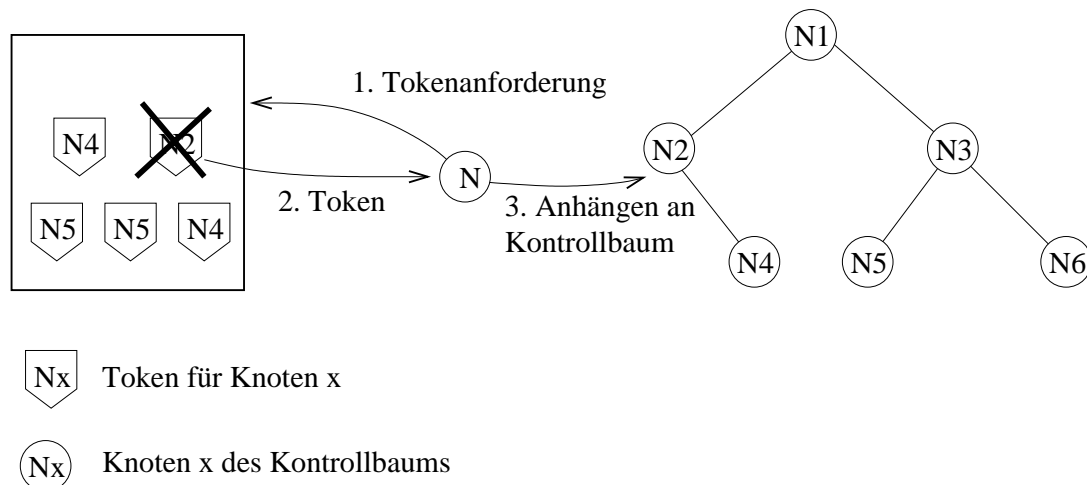


Abbildung 5.2.3: Token Repository Service

Den *Token Repository Service* gibt es in verschiedenen Varianten: Sie unterscheiden sich in der Art, wie TRS ein Token für eine bestimmte Anfrage aus seinem Repository auswählt. Beim *Token Repository Service with Random Choice (TRS-RC)* wird das Token zufällig ausgewählt, beim *Token Repository Service with Minimum Height (TRS-MH)* wird das Token so ausgewählt, dass ein möglichst höhen-balancierter Kontrollbaum entsteht [Maihöfer99].

Um nicht die Skalierbarkeitsprobleme eines einzelnen TRS-Servers zu haben, ist TRS als verteilter Dienst mit einer Hierarchie von Servern realisiert, was zu einer guten Skalierbarkeit führt. Die Server sind hierzu in einer Baumstruktur angeordnet: Wenn ein lokaler TRS-Server kein geeignetes Token liefern kann, wird in diesem Baum der verteilten Server nach einem geeigneten Token gesucht. Anders ist die Verteilung bei einer weiteren Variante von TRS, dem *Token Repository Service with Proxy Server Strategy (TRS-PS)* realisiert: Hier sind die Server nicht in einer Baumstruktur angeordnet. Ist

hier beim lokalen TRS-Server kein geeignetes Token vorhanden, so wird mit ERS (siehe 5.2.1) ein Server gesucht, der ein geeignetes Token zur Verfügung hat. Dieser Ansatz hat den Vorteil, dass ein Token gesucht wird, dessen Eigentümer so nahe wie möglich bei dem neuen Gruppenmitglied ist, das sich an den Kontrollbaum anhängen möchte.

Im Gegensatz zu ERS und ERA ist der TRS nicht vom eingesetzten Routingprotokoll abhängig. Im Gegensatz zu ERA gibt es auch keinen Nachrichtenaufwand, wenn sich kein neuer Knoten anmelden möchten. Zu beachten ist auch, dass für das beschriebene Verfahren nur Unicast-Nachrichten benötigt werden.

5.3 Protokolle

5.3.1 Log-Based Receiver Reliable Multicast (LBRM)

Als Beispiel für ein Transportprotokoll, das versucht, aufbauend auf dem IP-Multicast, mit einem reinen *empfängerinitiierten* Ansatz Zuverlässigkeit zu garantieren, soll der *Log-Based Receiver-reliable Multicast (LBRM)* [Holbrook99] dienen.

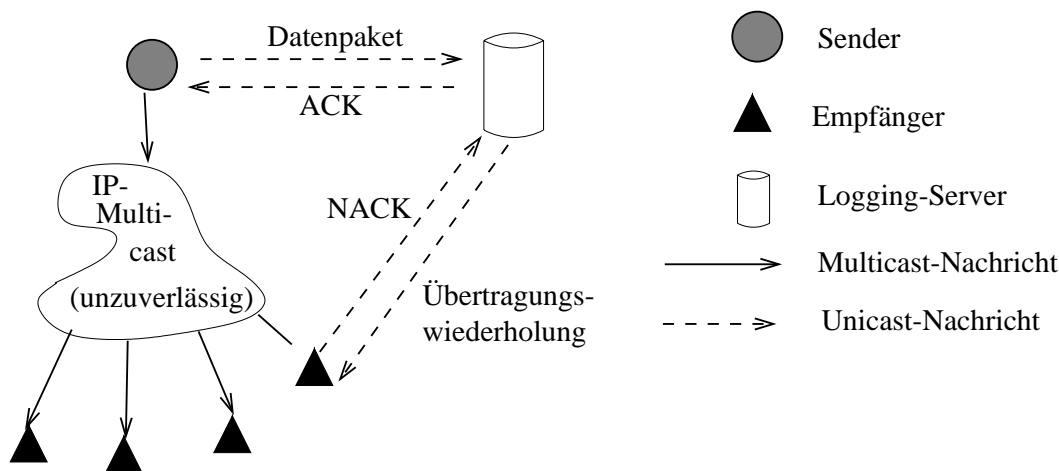


Abbildung 5.3.1: Funktionsweise von LBRM

Wie in 4.6 ausgeführt, benötigt ein reines empfängerinitiiertes Verfahren unendlichen Speicher für absolute Zuverlässigkeit. LBRM verwendet daher einen so genannten *Logging-Server*, der alle über Multicast versendeten Pakete speichert. Er ist den Anforderungen der Anwendung entsprechend großzügig mit Speicher (in der Regel auch persistentem Speicher) ausgestattet. Der Sender schickt alle Nachrichten, die er über Multicast verschickt, auch über Unicast direkt an den Logging-Server. Sobald dieser den Empfang mit einem ACK bestätigt, kann der Sender den Speicher freigeben. Die Empfänger fordern Übertragungswiederholungen beim Logging-Server an (siehe Abbildung 5.3.1).

Da dieser Ansatz anfällig für eine NAK-Impllosion beim Logging-Server ist, wurde auch eine Version mit einer Hierarchie von Logging-Servern vorgeschlagen.

Der Nachteil dieses Ansatzes ist natürlich der große Aufwand für die Logging-Server und das Problem, dass es schwer ist, den nötigen Speicher zu bestimmen. Ein Vorteil des Verfahrens ist, dass auch Empfänger, die der Gruppe erst später beitreten, noch die vollständigen Daten bekommen können.

5.3.2 Scalable Reliable Multicast Protocol (SRM)

Der *Scalable Reliable Multicast (SRM)* [Floyd97] ist in das im *MBone* (siehe 3.4.4) eingesetzte *wb* (von engl.: *whiteboard*), eine Anwendung für verteilte Gruppenarbeit, integriert. Auch bei SRM handelt es sich um ein rein empfängerbasiertes Verfahren. Hier muss die Anwendung in der Lage sein, alle Pakete bei Bedarf wiederholt zu versenden. SRM verwendet keinen Kontrollbaum, sondern ein reines RINA-Verfahren. Anders als beim in 4.8.1 beschriebenen RINA-Verfahren wird bei einem festgestellten Paketverlust bei SRM das NAK nicht um eine rein zufällige Wartezeit verzögert: Ein Empfänger berechnet mit Hilfe von Statusnachrichten die Entfernung¹ zu seinen Nachbarn im Routingbaum und bezieht diese Information in die Wartezeit für das Senden des NAK ein: Im Prinzip wartet der Empfänger, ob nicht schon sein Vorgänger das Paket vermisst, und sendet erst dann sein NAK. Wie bei RINA beschrieben, wird das eigene NAK durch den Empfang des entsprechenden NAK eines anderen Empfängers unterdrückt.

5.3.3 TMTP

Nachdem drei Verfahren zum Aufbau von Kontrollbäumen vorgestellt worden sind, soll jetzt auf ein vollständiges zuverlässiges Multicast Transportprotokoll eingegangen werden. *Tree-Based Multicast Transport Protocol (TMTP)* [Yavatkar95] setzt auf dem IP Multicastprotokoll (siehe 3.4) auf und bietet der Anwendung zuverlässige Auslieferung und Flusskontrolle. Der Aufbau des Kontrollbaums erfolgt wie oben (5.2.1) beschrieben mit ERS.

Ein zentrales Element bei TMTP sind die so genannten *Domain Manager (DM)*: Sie sind die Knoten im Kontrollbaum. Jeder DM ist in seiner *Domain* für Übertragungswiederholungen und Flusssteuerung zuständig. Zu einer jeweiligen Domain gehören jeweils ein Vaterknoten und seine direkten Kinder (siehe Abbildung 5.3.2).

TMTP benutzt zur Fehlerkorrektur sowohl einen *senderinitiierten* als auch einen *empfängerinitiierten* Ansatz (siehe auch 4.5). Stellt ein Empfänger einen Paketverlust fest, so sendet er ein NAK an seinen DM. Hat dieser das Paket empfangen, so sendet er sofort eine Übertragungswiederholung mit einem begrenzten Multicast (mit einer geeigneten TTL, so dass er seine eigene Domain abdeckt). Hat der DM das Paket auch nicht empfangen, wendet er sich an seinen DM.

¹Genauer: die halbe *Round Trip Time* (RTT). Sie ist ein Maß für die Verzögerung zwischen zwei Knoten im Netzwerk. Die RTT zwischen Knoten A und Knoten B ist die Zeit, die ein Paket braucht, um von A nach B und wieder zurück nach A zu gelangen.

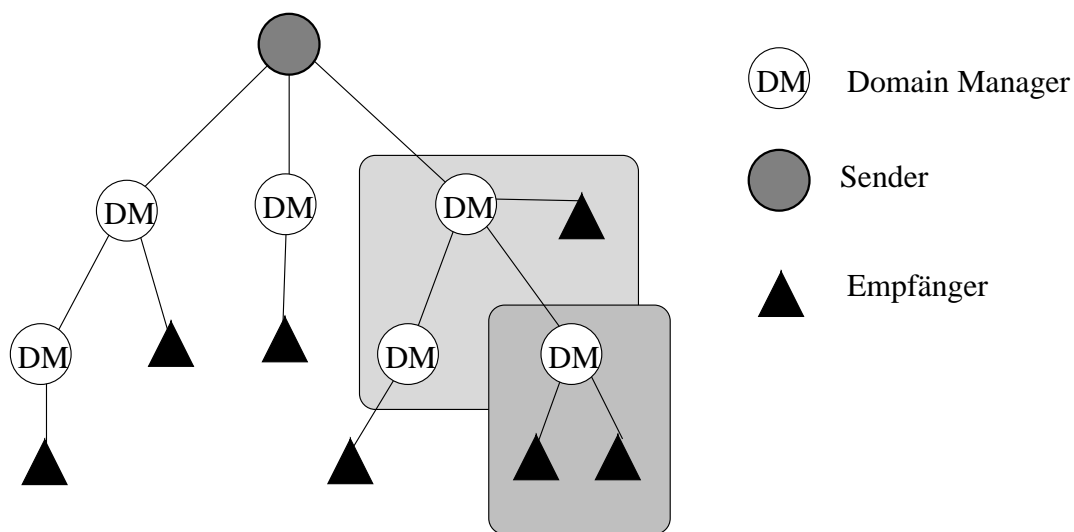


Abbildung 5.3.2: Kontrollbaum von TMTP

Zusätzlich hierzu verwendet TMTP akkumulierte ACKs. Ein Empfänger sendet sein ACK an seinen DM, welcher ein akkumuliertes ACK an seinen DM sendet. Wenn nach einer bestimmten Zeit kein akkumuliertes ACK beim Sender ankommt, nimmt dieser eine Übertragungswiederholung über Multicast an die ganze Gruppe vor. Optional können die DMs auch sofort ein ACK verschicken, sobald sie ein Paket empfangen haben. Sie müssen das Paket dann allerdings so lange speichern, bis sie von allen Nachfolgerknoten ein ACK empfangen haben.

Die Flusssteuerung basiert auf einer maximalen Senderate und einem Fenstermechanismus. Entsprechend dem Fenstermechanismus bei Unicast-Protokollen wird das Fenster beim Empfang eines akkumulierten ACKs vorgerückt.

Kapitel 6

Router-basierte Verfahren

In diesem Kapitel wird ein Überblick über die so genannten *Router-basierten* Verfahren gegeben. Einige existierende Protokolle werden vorgestellt. Am Ende des Kapitels folgt noch eine Zusammenfassung.

Inhaltsangabe

6.1	Überblick über Router-basierte Ansätze	40
6.2	Lightweight Multicast Services (LMS)	40
6.3	OTERS	45
6.4	Active Reliable Multicast (ARM)	50
6.5	Reliable Multicast Architecture (RMA)	52
6.6	Pragmatic General Multicast (PGM)	58
6.7	Weitere Verfahren	64
6.8	Prinzipien der Router-basierten Ansätze	64

6.1 Überblick über Router-basierte Ansätze

Ein Hauptproblem beim Entwurf von zuverlässigen Multicastverfahren ist die Schwierigkeit, an Informationen über die Netzwerktopologie zu gelangen. Diese Informationen sind für Fehlerkontrollverfahren sehr nützlich. So bedeutet beim Multicast der Verlust eines Pakets, dass ein ganzer Unterbaum, vom Punkt des Verlusts aus gesehen, betroffen ist. Eine naheliegende Lösung wäre also, dass ein direkt unter dem Punkt des Verlusts gelegener Knoten einen Request (NAK) direkt an den Knoten über dem Verlust schickt und dieser das fehlende Paket direkt per Multicast an den betroffenen Unterbaum schickt. Topologieinformation ist jedoch in den Routern (Schicht 3) vorhanden und ist an den Endpunkten der Multicastkommunikation (Schicht 4) nicht ohne weiteres verfügbar. Router an der Fehlerkontrolle zu beteiligen, wurde daher bisher oft als Verletzung des Schichtenmodells und im Widerspruch zu den *End-To-End Argumenten* [Saltzer84] gesehen. Um effiziente Verfahren zur Bewältigung der oben (siehe 4.7) genannten Skalierbarkeitsprobleme implementieren zu können, bietet es sich dennoch an, einige Dienste auf Schicht 3 zu implementieren. Ansätze, die auf diese Dienste aufbauen, nennt man *Router-basierte Verfahren*.

6.2 Lightweight Multicast Services (LMS)

6.2.1 Überblick

LMS [Papadopoulos98] hat als Ausgangspunkt die oben erwähnte naheliegende Lösung für einen Paketverlust beim Multicast. Wie Abbildung 6.2.1 zeigt, ist vom Verlust eines Multicastpakets ein ganzer Unterbaum betroffen. Die Wurzel dieses Unterbaums ist der Punkt, an dem der Paketverlust aufgetreten ist.

In Abbildung 6.2.2 sieht man die optimale Lösung:

1. der Empfänger direkt unter dem Punkt des Verlusts sendet ein NAK an den Empfänger direkt über dem Punkt des Verlusts
2. dieser Empfänger sendet das fehlende Paket über Multicast an den betroffenen Unterbaum.

Diese Lösung wird durch drei Elemente ermöglicht: Die in den Routern vorhandene Information wird verwendet, um einen so genannten *Replier* zu bestimmen, der für die Beantwortung von Requests der Empfänger in einem Unterbaum zuständig ist. Zweitens wird ein *Turning Point* als der Punkt im Baum definiert, an dem ein Request, der sich nach oben bewegt, wieder nach unten in Richtung *Replier* befördert wird (siehe Abbildung 6.2.2). Drittens führt der Router am *Turning Point* einen *Subcast*¹ durch

¹Kurzform von „Subtree Multicast“

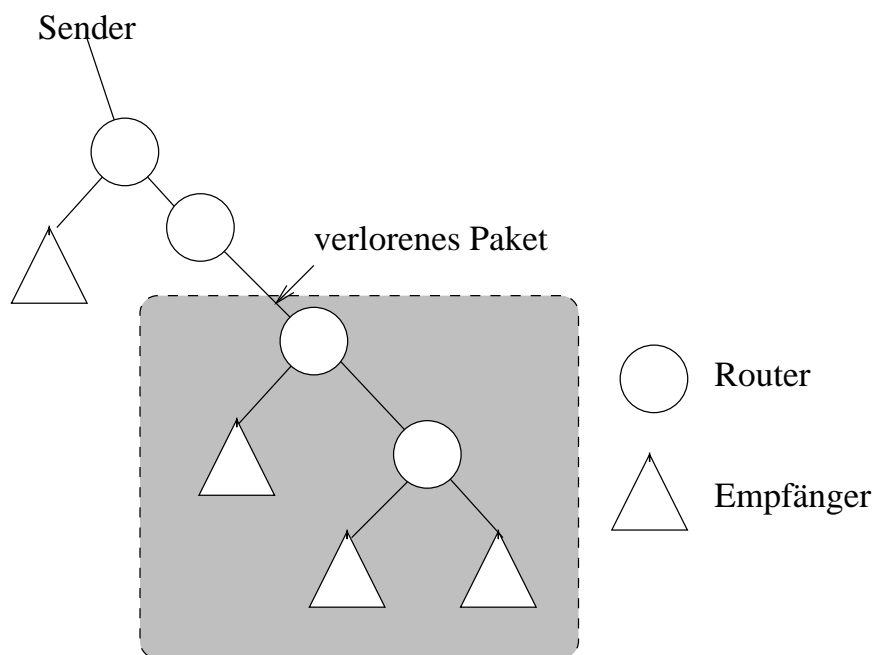


Abbildung 6.2.1: Verlust eines Pakets beim Multicast

(sendet das verlorene Paket an den durch den *Turning Point* definierten Unterbaum). Vorausgesetzt wird hier, dass es sich beim Routing Protokoll um ein *Source Routed Tree* Verfahren (z.B. DVMRP, siehe 3.3.2) und nicht um ein *Shared Tree* Verfahren (z.B. CBT, siehe 3.3.3) handelt, sonst sind Änderungen am Verfahren nötig.

6.2.2 Repliers

Jeder Router kennt genau einen *Replier*. Dabei wird nicht die Adresse des zuständigen *Repliers* gespeichert, sondern nur der Link, der zum *Replier* führt. Jeder Router wählt einen *Replier-Link* aus seinen Nachfolgerknoten (*Downstream-Links*) im Multicast Routingbaum aus. Eine Ausnahme bildet der Fall, dass es nur einen Nachfolgerknoten gibt, dann wird der Vorgängerknoten (*Upstream-Link*) gewählt. Auch der Router, der direkt mit dem Sender verbunden ist, bildet eine Ausnahme, er wählt immer die Verbindung zum Sender als *Replier-Link* (siehe Abbildung 6.2.3).

Die Unterscheidung zwischen *Upstream* und *Downstream* ist schon für die Routingprotokolle notwendig und liegt daher in den Routingtabellen vor, stellt also keinen zusätzlichen Aufwand dar. Zusätzlich muss noch die Information über den *Replier-Link* in den Routingtabellen gehalten werden. Für das Verfahren sind im einzelnen folgende Statusinformationen im Router notwendig²:

- der *Upstream-Link*
- eine Liste der *Downstream-Links*

²jeweils pro Sender und pro Multicastgruppe

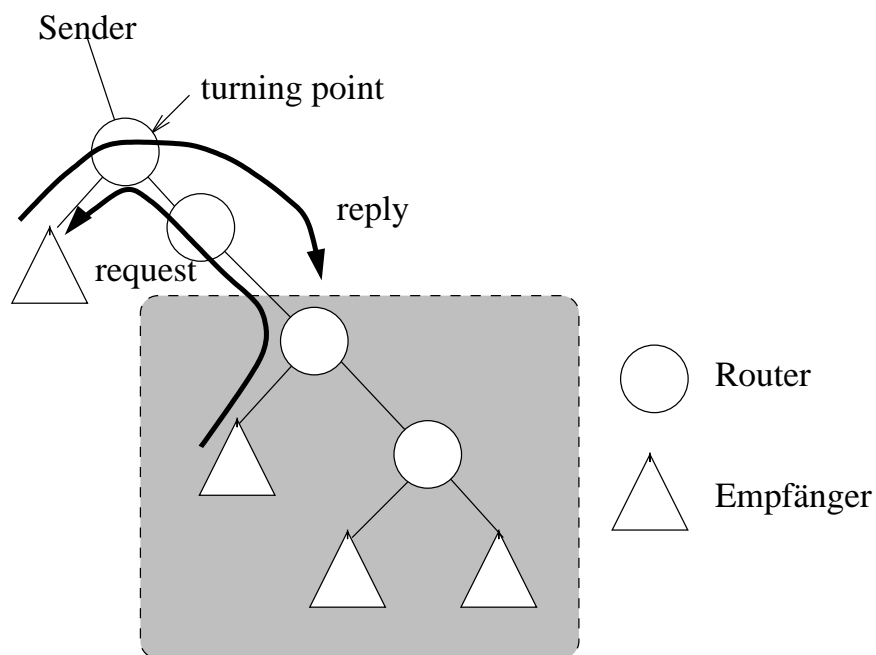


Abbildung 6.2.2: optimale Fehlerkorrektur

- der *Replier-Link*
- die Kosten, um den aktuellen *Replier* zu erreichen (z.B. Hops oder Fehlerrate)
- ein Timer für den Replier-Eintrag

Wie erwähnt sind die Einträge für den *Replier-Link* und für die Kosten, um den *Replier* zu erreichen der einzige zusätzliche Aufwand. Die Statusinformationen über *Replier* werden wie die Gruppenmitgliedschaftsinformationen mit Hilfe von Kontrollnachrichten (z.B. IGMP, siehe auch 3.1.3) verbreitet. Zwei neue Arten von Kontrollnachrichten sind dafür nötig:

Want_to_be_replier: Empfänger, die bereit sind, die Rolle eines Repliers zu übernehmen, senden diese Kontrollnachricht regelmäßig an die Router. In der Nachricht sind die Kosten für den Replier enthalten, so dass die Router den Replier mit den geringsten Kosten wählen können. Der Router merkt sich den Link, auf dem die *Want_to_be_replier* Nachricht empfangen wurde. Sind die Kosten der empfangenen Nachricht höher als die Kosten des aktuellen *Repliers*, wird die Nachricht ignoriert. Wenn die Nachricht über den *Replier-Link* eintrifft und die Kosten gleich den aktuellen Kosten sind, wird für den aktuellen Eintrag der Timer zurückgesetzt. Sind die Kosten der empfangenen Nachricht geringer als die aktuellen Kosten, wird der *Replier-Link* Eintrag in der Routingtabelle entsprechend geändert.

Replier_gone: Wenn ein Empfänger, der eine *Want_to_be_replier* Nachricht verschickt hat, die Multicast-Gruppe verlässt, informiert er die Router durch diese Nachricht. Diese Nachricht ist nicht zwingend notwendig, da die *Replier* Information

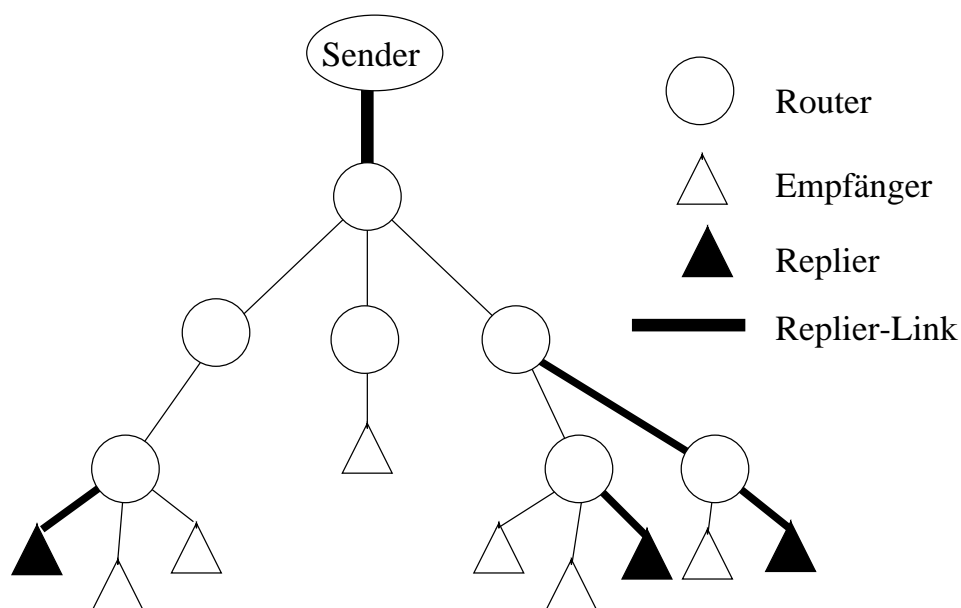


Abbildung 6.2.3: Zuordnung von Repliern zu Routern

in den Routern *Soft State*³ ist. Empfängt ein Router auf seinem *Replier-Link* eine *Replier_gone* Nachricht, so wird der *Replier-Link* Eintrag in der Routingtabelle gelöscht. Bis auf weiteres fungiert nun der Vorgängerknoten (Upstream) als Replier. Als Optimierung kann sich der Router jeweils auch den zweitbesten Replier merken, so dass dieser dann verwendet werden kann. Wird die *Replier_gone* Nachricht auf einem anderen Link als dem *Replier-Link* empfangen, wird sie ignoriert und weggeworfen.

6.2.3 Behandlung von NAKs

Hier handelt es sich um einen empfängerbasierten Ansatz: Bemerkt ein Empfänger ein fehlendes Paket, sendet er ein NAK. Die Router sind dafür verantwortlich, dieses NAK zu einem (möglichst optimalen) *Replier* zu befördern. Hierfür wird der neue Kontrollnachrichtentyp *Forward_to_replier* eingeführt, der durch eine eigene IP-Option im Multicastheader erkannt und von jedem Router ausgewertet wird. Der neue Kontrollnachrichtentyp enthält folgende Informationen:

- zur Identifizierung des zuständigen Routingbaumes wird das Tupel

`<Sender, Multicastadresse>`

verwendet

³Nach Ablauf des oben erwähnten Timer wird sie verworfen

- Das Tupel

<Routeradresse, Link-ID>

ist ursprünglich (beim Absenden des NAKs) leer und wird zum Festhalten des *Turning Point* verwendet.

Jeder Router, der eine *Forward_to_replier* Nachricht empfängt, untersucht die enthaltenen Kontrollinformationen und verfährt entsprechend:

- Ist die Multicastadresse unbekannt, so wird die Nachricht weggeworfen.
- Kommt die Nachricht von einem Nachfolgerknoten (Downstream) und der empfangende Router hat keinen *Replier-Link*, dann wird die Nachricht mit unveränderter Kontrollinformation zum Vorgängerknoten (Upstream) weitergereicht (siehe Abbildung 6.2.4 (a)). Ebenso wird verfahren, wenn die Nachricht auf dem *Replier-Link* des Routers eintrifft.
- Gibt es einen *Replier-Link* und die Nachricht trifft auf einem anderen Nachfolgerknoten ein, so hat die Nachricht ihren *Turning Point* erreicht. In der Kontrollinformation wird das Tupel

<Routeradresse, Link-ID>

eingefügt und die Nachricht auf dem *Replier-Link* weitergeleitet (siehe Abbildung 6.2.4 (b)). Routeradresse bezeichnet hier die Unicast-Adresse des Routers und Link-ID ist die Bezeichnung des Netzwerkinterface, auf dem die Nachricht von diesem Router empfangen wurde.

- Trifft die Nachricht von einem Vorgängerknoten (Upstream) ein und es existiert ein *Replier-Link* und das Tupel

<Routeradresse, Link-ID>

ist nicht leer, dann wird die Nachricht unverändert auf dem *Replier-Link* weitergeleitet (siehe Abbildung 6.2.4 (c)). Ist das Tupel leer, so wird die Nachricht weggeworfen. Gibt es keinen *Replier-Link*, wird ein Fehler signalisiert (ein Router weiter oben im Routingbaum hat die falsche Information, dass dieser Pfad zu einem *Replier* führt).

6.2.4 Antwort über Subcast

Bisher wurde beschrieben, wie *Replier* gewählt und NAKs zu ihnen geroutet werden. Erreicht nun einen *Replier* ein NAK, so nutzt er den *Subcast* Dienst, um die Daten an den richtigen Unterbaum zu senden. Ein Subcast besteht aus zwei Teilen: Über Unicast wird die Nachricht an einen Router geschickt. Dieser Router schickt die Nachricht über

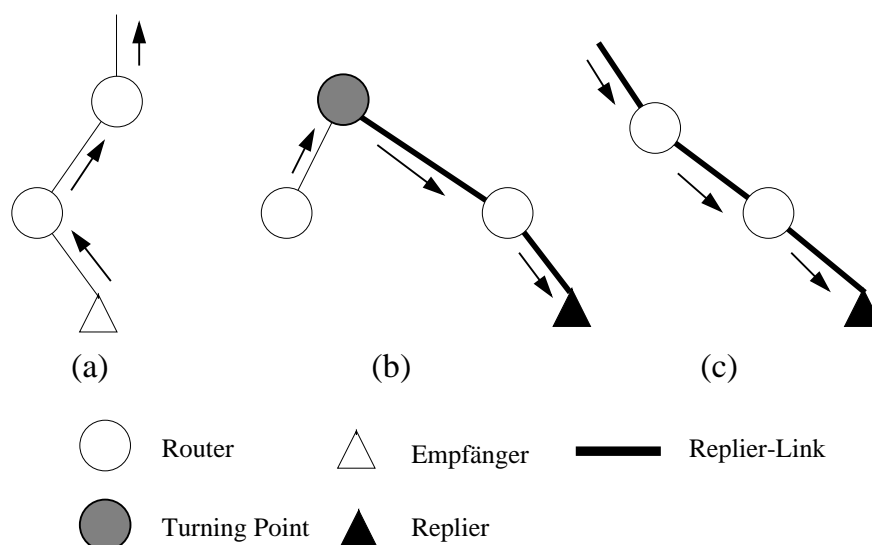


Abbildung 6.2.4: Behandlung von *Forward_to_Replier* Nachrichten an Routern

Multicast an einen seiner Nachfolger (auf nur einem Downstream-Link) weiter. Der Subcast erreicht also nur einen Unterbaum mit dem Nachfolger des genannten Routers als Wurzel. Im einzelnen läuft hierbei folgendes ab (siehe auch Abbildung 6.2.5):

1. Der Replier empfängt eine *Forward_to_replier* Nachricht, die wie oben beschrieben Informationen über den *Turning Point* enthält ($\langle \text{Routeradresse}, \text{Link-ID} \rangle$).
2. Der Replier sucht in seinem Pufferspeicher das gewünschte Paket. Findet er es nicht, so leitet er den Request mit einer neuen *Forward_to_replier* Nachricht weiter, setzt aber die *Turning Point* Information der empfangenen Nachricht ein. Damit ist sichergestellt, dass der neue Replier die Antwort an den richtigen Unterbaum sendet.
3. Findet der Replier das gewünschte Paket im Puffer, so erzeugt er ein Multicastpaket, das die gewünschten Daten enthält. Das Multicastpaket wird dann per Unicast⁴ an den *Turning Point* Router geschickt.
4. Der Router am *Turning Point* packt das Multicastpaket aus und sendet es per Multicast über den spezifizierten Link.

6.3 OTERS

6.3.1 Überblick

Wie das oben beschriebene *LMS* verwendet auch *On-Tree Efficient Recovery using Subcasting (OTERS)* [Li98] den gleichen Ansatz: Mit Hilfe von Erweiterungen in der

⁴Analog zum *tunneling* im Mbone (siehe auch 3.4.4).

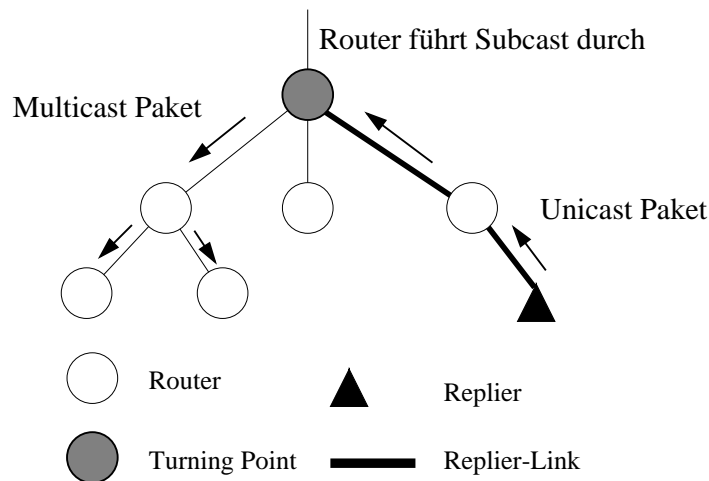


Abbildung 6.2.5: ein Replier antwortet über Subcast

Netzwerkschicht wird die Multicast-Routinginformation zur Verhinderung der NAK Implosion und zur optimalen Fehlerkorrektur verwendet. Es wird hierzu ein so genannter *NAK Fusion Tree* aufgebaut, der dem Multicast Routingbaum entspricht. Auch hier wird ein *Subcast* Mechanismus verwendet, der sich allerdings leicht von dem in *LMS* eingesetzten unterscheidet. *OTERS* gliedert sich in zwei Teilprotokolle. Das *Fusion Tree Formation Protocol (FTFP)* baut eine Hierarchie aus Unterbäumen (deren Wurzel gewisse Ähnlichkeit mit den *Turning Points* in *LMS* haben) und dafür zuständigen *Designated Receivers (DR)* (zum Teil vergleichbar mit *Replier* bei *LMS*) auf. Das zweite Protokoll ist das *Loss Recovery Protocol (LRP)*, das für die Beförderung der NAKs und das Senden der fehlenden Pakete an den entsprechenden Unterbaum zuständig ist. Im folgenden wird zunächst *FTFP* und die dafür nötigen Erweiterungen besprochen, anschließend wird auf *LRP* eingegangen.

6.3.2 Das Fusion Tree Formation Protocol (FTFP)

Für das bessere Verständnis von *FTFP* müssen erst einmal einige Begriffe eingeführt werden (siehe auch Abbildung 6.3.1): Der Multicast Routingbaum wird in so genannte Untergruppen unterteilt. Jede Untergruppe hat eine Unterwurzel (engl. *Subroot*) und einen *Designated Receiver (DR)*, der Mitglied der Untergruppe ist und sich nahe an der Unterwurzel befindet.

Das Konzept ist insofern hierarchisch bzw. rekursiv, als ein DR sowohl als DR für die Mitglieder seiner Untergruppe als auch als DR für seine Untergruppen fungiert. Jedes Gruppenmitglied kennt seinen DR, jeder DR kennt mindestens seinen DR und seine Unterwurzel. Die so genannte „oberste Unterwurzel“ eines DR ist die Unterwurzel, die am nächsten am Sender liegt. Die Suche eines Gruppenmitglieds nach seinem DR läuft nun folgendermaßen ab:

1. Suche nach einer möglichen Unterwurzel.

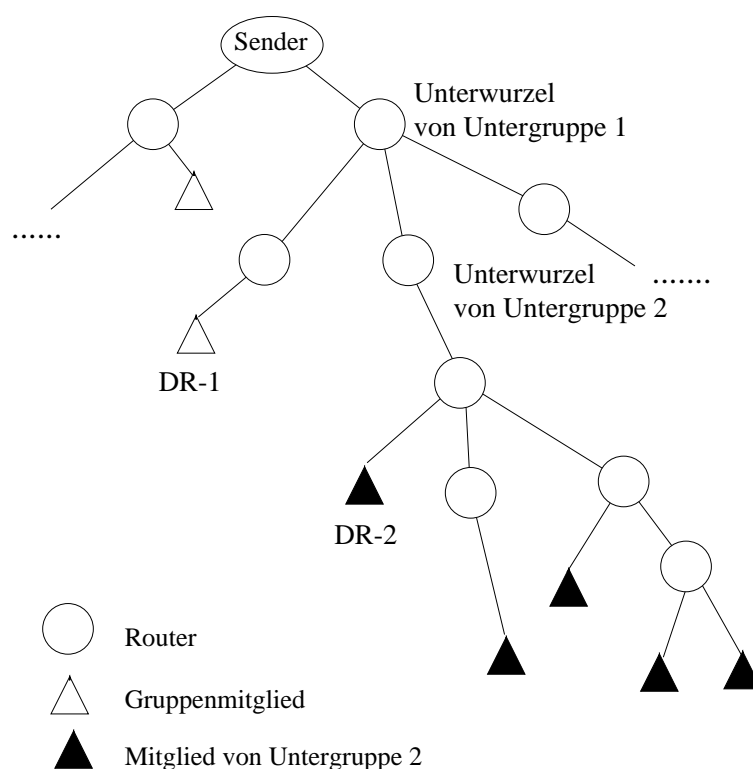


Abbildung 6.3.1: ein einfacher OTERS Fusion Tree

2. Sende über diese Unterwurzel per *Subcast* ein *FTFP Paket* an die Untergruppe dieser Unterwurzel und warte eine gewisse Zeit.
3. Wird in dieser Zeit ein *FTFP Paket* empfangen, das von einem als DR geeigneteren Mitglied kommt als vom Initiator, wird dieses als DR festgehalten, wenn nicht, wird der Initiator selbst zum DR für die Untergruppe.

Im folgenden sollen die einzelnen Schritte genauer untersucht werden: Bei der Suche nach einer geeigneten Unterwurzel wird eine der notwendigen Erweiterungen der Netzwerkschicht, *Multicast Route Backtracing*, verwendet. Diese erlaubt es dem Mitglied einer Multicastgruppe, die Folge der Router im Routingbaum zwischen sich selbst und dem Sender zu ermitteln. Eine solche Anfrage enthält den Absender M , die Gruppenadresse G , den Sender S und die Obergrenze der Hops K . Die Antwort enthält eine Liste der IP-Adressen der K letzten Hops auf dem Pfad von S nach M . Die Beschränkung auf K Hops ermöglicht eine inkrementelle Suche und verhindert eine Nachrichtenimplosion beim Sender. Die beschriebene Funktionalität kann als eine Untermenge der in *IGMP Traceroute (Mtrace)* [Fenner97] beschriebenen Funktionalität implementiert werden.⁵ Bei der Suche nach einer möglichen Unterwurzel beginnt *FTFP* mit $K = 1$ und inkrementiert K bis eine Unterwurzel R_K gefunden wird. Wie in Punkt 2 angeführt, sendet nun M einen *Subcast* über die gefundene Unterwurzel R_K . Hier kommt

⁵*Mtrace* selbst ist für *FTFP* auch problemlos verwendbar, kann aber aus Sicherheits- und Datenschutzgründen unter Umständen nicht im ganzen Netzwerk zur Verfügung gestellt werden.

die zweite erforderliche Erweiterung der Netzwerkschicht zum Einsatz. Die in *OTERS* verwendete Variante des *Subcasting* unterscheidet sich von der in *LMS* beschriebenen. Bei *OTERS* wird der *Subcast* an alle Nachfolgerknoten der Unterwurzel ausgeliefert, bei *LMS* nur auf einem Link. Auch die Kapselung unterscheidet sich von der in *LMS*. In *OTERS* wird folgender Mechanismus verwendet:

Sei

$$(\text{Sender, Empfänger})[\text{Nutzdaten}]$$

die Notation für ein IP-Paket, dann ist ein *OTERS* Subcast Paket folgendermaßen aufgebaut (Absender M , Router R , Multicastgruppe G , Sender S):

$$(\text{M,R})[(\text{S,G})[\text{Nutzdaten}]]$$

M sendet das Paket per Unicast an R . R vertauscht dann die inneren mit den äußeren IP-Headern. Das resultierende Paket

$$(\text{S,G})[(\text{M,R})[\text{Nutzdaten}]]$$

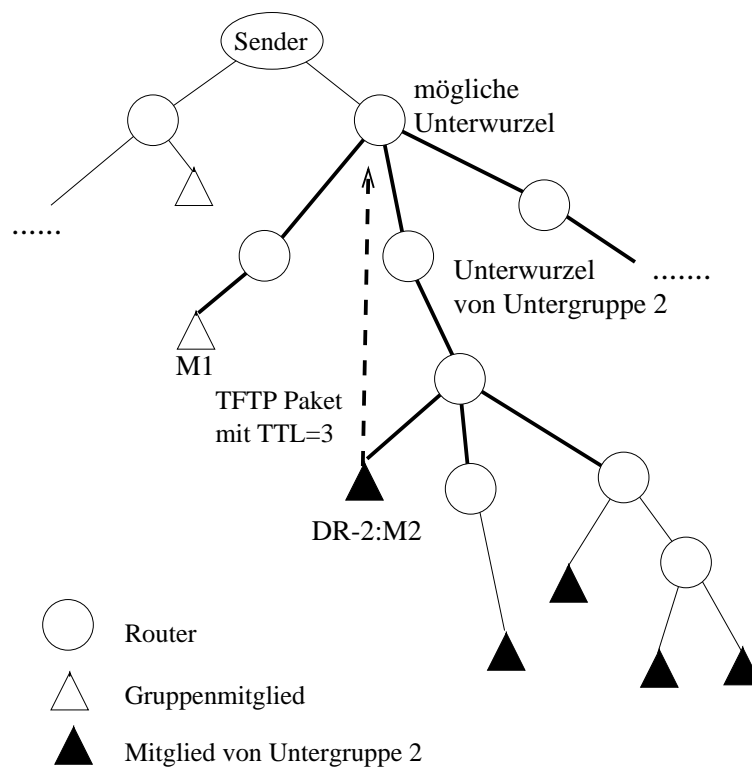
wird dann von R wie ein normales IP-Multicastpaket weitergeleitet. Dieses Verfahren hat gegenüber dem in *LMS* verwendeten Verfahren den Vorteil, dass der Empfänger eines Subcast Paketes den Absender überprüfen kann.

Ein *FTFP* Paket von M über R_K per *Subcast* enthält dabei folgende Daten: Die Hops K zwischen M und R_K , die Verzögerung D zwischen Sender und M , die Verlustrate L bei M und der ursprüngliche TTL-Wert für den *Subcast*.

Nun zurück zu Punkt 2: M hat einen *Subcast* über R_K geschickt. Sobald dieses *FTFP* Paket von M selbst wieder empfangen wird, startet M einen Timer mit dem doppelten Wert der Verzögerung D . Der geeigneter DR wird anhand eines alphabetischen Vergleichs der Tupel (L,D,K) der während des laufenden Timers eintreffenden *FTFP* Pakete bestimmt (Punkt 3).

Ein DR verschickt regelmäßige *FTFP* Pakete über seine oberste Unterwurzel (s.o.). Wenn ein Untergruppenmitglied durch fehlende *FTFP* Pakete feststellt, dass sein DR nicht mehr da ist, so wird entweder der nächst beste genommen oder, wie oben geschildert, die aktive Suche wieder gestartet. Empfängt ein Gruppenmitglied M über Unterwurzel R eine *FTFP* Nachricht, so gibt es folgende Möglichkeiten:

- die Nachricht wird ignoriert, wenn M schon einen besseren DR hat
- wenn M ein besserer DR für die Untergruppe ist, sendet M eine eigene *FTFP* Subcastnachricht über R

Abbildung 6.3.2: Beispiel für *FTFP*

- sonst wird der Absender der *FTFP* Nachricht als neuer DR bei *M* eingetragen

Abbildung 6.3.2 zeigt den Fall, dass M_2 sich an der Bildung einer Untergruppe beteiligt. M_2 hat eine mögliche Unterwurzel gefunden (mtrace mit Hop-Limit $k = 3$) und sendet nun über *Subcast* ein *FTFP* Paket (ebenso mit $TTL = 3$, was verhindert, dass die unteren Teile des Baumes erreicht werden⁶). Im angegebenen Beispiel ist zu erwarten, dass M_1 einen besseren DR darstellt und daher ein eigenes *FTFP* Paket verschickt, worauf M_2 M_1 als DR nimmt.

6.3.3 Das Loss Recovery Protocol (LRP)

Auch bei *OTERS* wird wie bei *LMS* von einem empfängerbasierten Ansatz ausgegangen: Bemerkte ein Empfänger (im folgenden E genannt) ein fehlendes Paket, sendet er ein NAK über Unicast direkt an seinen DR. Ist E selbst ein DR, so verlangt er, dass das fehlende Paket per *Subcast* von seiner obersten Unterwurzel aus verschickt wird. Handelt es sich bei E nicht um einen DR, soll ihm das Paket direkt per Unicast zugeschickt werden. Folgende Möglichkeiten gibt es bei der Ankunft eines NAK bei einem DR:

⁶Stellt eine Optimierung dar. Das Protokoll ist auch ohne Verwendung der TTL korrekt.

- der DR hat das Paket selbst nicht und hat auch noch kein NAK gesendet
 \implies der DR schickt ein NAK an seinen DR
- der DR hat das Paket selbst nicht, hat aber schon ein NAK an seinen DR gesendet
 \implies das empfangene NAK wird verworfen
- der DR hat vor kurzem das fehlende Paket als *Subcast* verschickt oder empfangen
 \implies das empfangene NAK wird verworfen
- sonst wird das fehlende Paket erneut verschickt

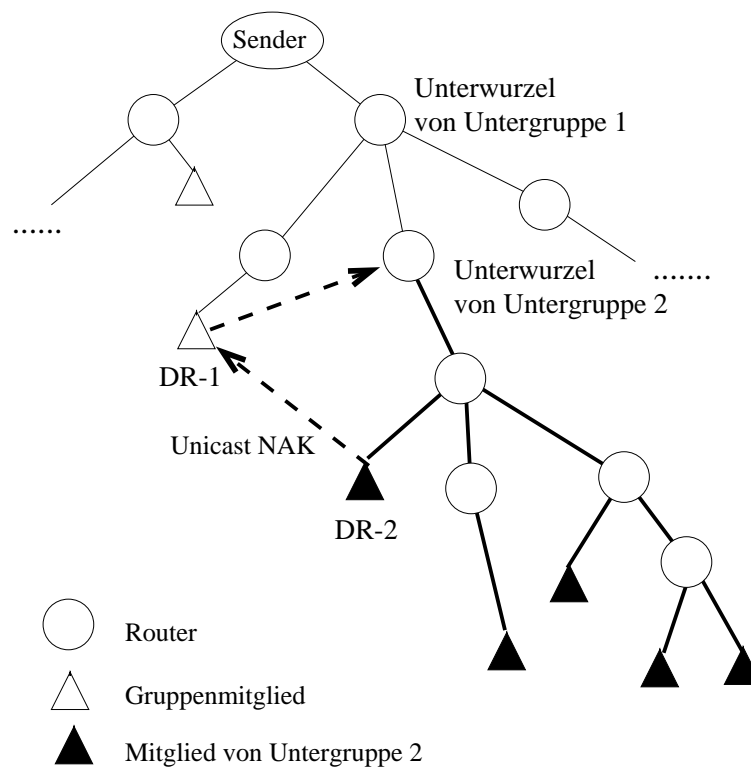


Abbildung 6.3.3: NAK und Fehlerkorrektur beim *LRP*

Abbildung 6.3.3 zeigt ein Beispiel: DR-2 schickt über Uicast ein NAK an seinen *Designated Receiver* DR-1. Dieser hat das gewünschte Paket empfangen und sendet (da der Sender des NAK selbst ein DR ist) das Paket über *Subcast* an die Unterwurzel der Untergruppe 2.

6.4 Active Reliable Multicast (ARM)

6.4.1 Überblick

Auch *ARM* [Lehman98] setzt wie die beiden bisher beschriebenen Verfahren auf Erweiterungen in der Netzwerkschicht, um NAK Implosion zu verhindern und eine möglichst

lokale Fehlerbehebung zu erreichen. Hat man jedoch bei *LMS* und *OTERS* versucht, die zusätzlich von den Routern zu übernehmenden Funktionen auf ein Minimum zu beschränken, so geht *ARM* von so genannten „Aktiven Knoten“ aus, die in erheblichem Umfang Speicherplatz und Rechenkapazität zur Verfügung stellen. Vorausgesetzt wird auch, dass die Pfade im Multicast Routingbaum den umgekehrten Unicast Routingpfaden entsprechen: *ARM* verlässt sich darauf, dass NAKs, die per Unicast vom Empfänger zum Sender geschickt werden, durch die gleichen Router laufen wie die entsprechenden Multicastpakete vom Sender zum Empfänger. In den folgenden drei Abschnitten werden die Funktionen der „aktiven Knoten“ genauer beschrieben. Das Protokoll ist so konzipiert, dass nicht alle Knoten „aktiv“ sein müssen - im Extremfall, wenn kein „aktiver Knoten“ vorhanden ist, wird aber weder NAK Implosion verhindert noch gibt es eine isolierte Fehlerbehebung.

6.4.2 Zwischenspeicherung von Multicastpaketen

In *ARM* werden Multicastpakete im Speicher der Router zwischengespeichert. Die Router unterhalten hierfür einen Cache. Der Header der Multicast Pakete wird hierfür um eine *Cache TTL* erweitert. Diese gibt an, wie lange das Paket sinnvollerweise im Speicher des Routers gehalten werden soll. Diese Zeit hängt davon ab, wie schnell ein Router mit dem NAK des am weitesten entfernten Empfängers rechnen kann. Dies hängt wiederum davon ab, wie schnell ein Empfänger einen Paketverlust feststellt und wie gross die Verzögerung (RTT) zwischen Router und Empfänger ist. Im Cache wird dann das Paket selbst, die genannte *Cache TTL*, die Gruppenadresse, der Sender und die Sequenznummer abgelegt, was eine eindeutige Identifizierung zulässt.

6.4.3 NAK Vermeidung und lokale Fehlerbehebung

Zusätzlich zu dem oben beschriebenen Cache für Multicast Datenpakete und der normalen Multicast Routinginformation speichert ein *ARM* Router *NAK Records* und *REPAIR Records*.

Ein *ARM* NAK Paket enthält die Adresse des Empfängers, der das NAK erzeugt hat, die Adresse des Senders (des fehlenden Pakets), die Gruppenadresse, die Sequenznummer des fehlenden Pakets, den NAK Zähler und eine Cache TTL. Der NAK Zähler wird vom Erzeuger des NAK bei jedem weiteren gesendeten NAK für dieses Paket inkrementiert.

Ein *NAK Record* wird im *ARM* Router angelegt, wenn zu einem empfangenen NAK weder das entsprechende Datenpaket im Cache, noch ein entsprechender *NAK Record* oder *REPAIR Record* vorhanden ist. Der *NAK Record* enthält den maximalen NAK Zähler, der für dieses Paket empfangen worden ist und ein *Subscription Bitmap*, das anzeigt, auf welchen Verbindungen NAKs für dieses Paket eingegangen sind (dies wird für das gezielte Weiterleiten der fehlenden Pakete verwendet, siehe 6.4.4).

Ein *REPAIR Record* für ein bestimmtes Datenpaket enthält einen Vektor, der für jeden Link anzeigt, ob darüber schon eine Antwort auf ein NAK und wenn ja mit welchem *NAK Zähler* gesendet wurde.

Empfängt ein *ARM* Router ein NAK auf Link k gibt es folgende Möglichkeiten (nicht vollständig, für den exakten Algorithmus siehe [Holbrook99]):

- Gibt es schon einen *REPAIR Record* für dieses Paket und ist der NAK Zähler für Link k größer oder gleich dem des eingetroffenen NAKs, so wird es verworfen.
- Ist kein *REPAIR Record* vorhanden, aber das gewünschte Paket im Cache vorhanden, so wird das Paket auf Link k gesendet und ein entsprechender *REPAIR Record* angelegt.
- Ist weder ein *REPAIR Record*, ein *NAK Record* noch das Datenpaket im Cache vorhanden, so wird das NAK in Richtung Sender weitergeleitet und ein entsprechender *NAK Record* erstellt (das für Link k zuständige Bit im *Subscription Bitmap* gesetzt).
- Gibt es schon einen *NAK Record* für dieses Paket und ist der NAK Zähler größer oder gleich dem des eingetroffenen Pakets, so wird das für Link k zuständige Bit im *Subscription Bitmap* gesetzt und das NAK verworfen.

6.4.4 Isolierte Fehlerbehebung

Die oben besprochenen *Subscription Bitmaps* bewirken wie der *Subcast* bei *OTERS* und *LMS*, dass verlorene Pakete möglichst nur an die Empfänger der Multicastgruppe ausgeliefert werden, denen diese Pakete auch fehlen. Findet ein *ARM* Router zu einem empfangenen Paket einen entsprechenden *NAK Record*, so wird das Paket nur auf den im *Subscription Bitmap* angegebenen Links weitergeleitet: So wird sichergestellt, dass nur die Empfänger, die ein NAK verschickt haben, das Paket bekommen.

6.5 Reliable Multicast Architecture (RMA)

6.5.1 Überblick

Die *Reliable Multicast Architecture* [Levine97] stellt ein Framework für zuverlässigen Multicast dar. *RMA* bietet Unterstützung für sender- und empfängerbasierte Verfahren (ACKs und NAKs). Wie bei *OTERS* und *LMS* wird durch zusätzliche Funktionen auf Schicht 3 erreicht, dass der Kontrollbaum dem Multicast-Routingbaum entspricht. Auch ein *Subcastmechanismus* steht zur Verfügung. *RMA* stellt im Gegensatz zu *OTERS* und *LMS* kein monolithisches Protokoll für zuverlässigen Multicast dar, sondern bietet Funktionen an, die genutzt werden können (oder auch nicht). *RMA*

baut auf dem *Addressable Internet Multicast (AIM)* auf. *AIM* erlaubt die Adressierung bestimmter Mitglieder einer Multicastgruppe. Im folgenden sollen zunächst die wichtigsten für *RMA* relevanten Aspekte von *AIM* beschrieben werden.

6.5.2 Addressable Internet Multicast (AIM)

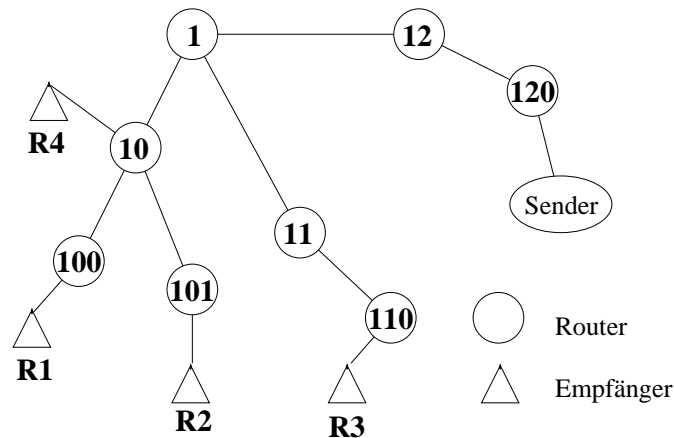
Ein Merkmal des IP-Multicast [Deering89] ist die Anonymität von Sendern und Empfängern: Ein Sender kann Daten an eine Multicastgruppe senden, ohne Informationen über Anzahl oder Adressen der einzelnen Empfänger zu benötigen. Auch Empfänger müssen nichts über andere Empfänger oder den Sender wissen. Die Stärke dieses Ansatzes ist, dass dadurch die Komplexität erheblich reduziert wird. IP-Multicast Adressen (Klasse D, siehe 3.4.2) sind eigentlich keine Adressen, sondern Namen für die Gruppe, die unabhängig von der Lokation der Gruppenmitglieder sind. Daher ist es nicht möglich, einzelne Pakete nur an eine Untergruppe der Multicastgruppe zu senden. Viele Protokolle verwenden deshalb das TTL-Feature von IP, um die Reichweite von Multicastpaketen zu beschränken (siehe 3.4.3). Auch die oben besprochenen Erweiterungen, wie zum Beispiel *Subcasting* oder *Multicast Route Backtracing*, gehen dieses Problem an. *AIM* löst das Problem auf grundlegende Weise: Die IP-Multicast Architektur wird um *gruppenrelative Adressinformationen* erweitert. Diese Erweiterung ist für alle Multicast Routingprotokolle, für Protokolle mit gemeinsamem Baum (engl. *shared tree*) oder senderbasiertem Baum (engl. *source-based tree*) möglich. *AIM* erweitert hierzu die Netzwerkschicht um drei Arten von *Labels*, die den Routern zugeordnet werden: *Positional Labels*, *Distance Labels* und *Stream Labels*. Mit Hilfe von *Stream Labels* können Untergruppen von Multicastgruppen gebildet werden. Da diese Funktionalität von *RMA* nicht genutzt wird, soll auf die *Stream Labels* im folgenden nicht näher eingegangen werden.

Positional Labels

AIM definiert eine Wurzel für jeden Multicast-Routingbaum. Ein *Positional Label* gibt die Position des Routers im Multicastbaum relativ zur (willkürlich gewählten) Wurzel an. Durch diesen *Label* hat jedes Mitglied der Multicastgruppe eine Adresse innerhalb des Routingbaums, mit der zum Beispiel ein Sender eine Untergruppe von Empfängern adressieren kann. Für diese Adressen wird eine Präfix-Notation verwendet:

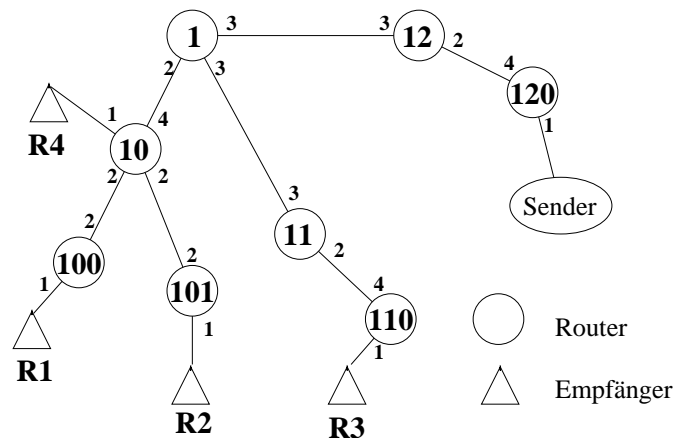
- Die Wurzel des Baums bekommt die Adresse 1.
- Alle direkten Nachfolgerknoten erhalten als *Label* das *Label* des Vorgängerknotens als Präfix und eine ganze Zahl, die unter den Geschwisterknoten eindeutig ist, als Suffix.

Abbildung 6.5.1 zeigt einen Routingbaum mit *Positional Labels*.

Abbildung 6.5.1: Routingbaum mit *Positional Labels*

Distance Labels

Im Gegensatz zu *Positional Labels*, die jeweils einem Router zugewiesen werden, bekommt jedes Interface (eines Routers), das zum Multicastbaum gehört, einen *Distance Label*. Es sind verschiedene Arten von *Distance Labels* vorgesehen. So Genannte „Typ 1“ *Distance Labels* geben jeweils die minimale Entfernung zu einem Router an, der mit einem Empfänger (der Gruppe) direkt verbunden ist. Die Metrik ist hier die Anzahl der Knoten (*hop count*). „Typ 1“ *Distance Labels* sind in AIM immer vorhanden, wobei für Anwendungen zusätzliche Typen definiert werden können, welche z.B. die Entfernung zu Gruppenmitglieder mit bestimmten Diensten angeben. Abbildung 6.5.2 zeigt einen Routingbaum mit *Distance Labels*.

Abbildung 6.5.2: Routingbaum mit *Distance Labels*

Positional Routing

Mit Hilfe der oben beschriebenen *Positional Labels* ist es möglich, Pakete an bestimmte Gruppenmitglieder zu schicken, ohne dass dazu Einträge in Routertabellen herangezogen werden müssen. Die *Labels* der Empfänger werden im Header des IP-Paketes

gespeichert. Router vergleichen das Label des Empfängers mit ihrem eigenen und leiten das Paket dementsprechend entweder an Vorgänger oder Nachfolger im Routingbaum weiter. Für das Beispiel in Abbildung 6.5.2 würde das so aussehen:

Beispiel: $R1$ (Pos. Label: 100) sendet Nachricht an Sender (Pos. Label: 120)

Es ergibt sich der Nachrichtenpfad:

$$R1 \rightarrow 100 \rightarrow 10 \rightarrow 1 \rightarrow 12 \rightarrow 120 \rightarrow \text{Sender}$$

Es ist auch möglich, eine Gruppe von Empfängern zu adressieren: Zum einen kann im Paket eine Liste von *Positional Labels* von Empfängern gespeichert werden, zum anderen lässt *AIM* die Verwendung von Masken zu. Bei der Verwendung von Masken wird die Nachricht an alle Router (und deren Empfänger) ausgeliefert, auf die die Maske passt. Bei Masken werden mit dem *Positional Label* zwei Flags (jeweils ein Bit) gespeichert. Ist das x Bit gesetzt, so wird die Nachricht an alle Router ausgeliefert, deren Präfix mit der Maske übereinstimmt. Ist das n Bit gesetzt, so wird die Nachricht an alle Router ausgeliefert, welche die Maske nicht als Präfix haben. Im Beispiel 6.5.2 würde eine Maske $10x$ die Auslieferung an den Unterbaum, der als Wurzel den Router 10 hat, bedeuten. Diese Funktionalität entspricht genau dem *Subcast* in *LMS* oder *OTERS*.

Anycast

Anycast liefert eine Nachricht an einen einzigen aus einer Menge von möglichen Empfängern aus (siehe auch 2.2.4). *Distance Labels* ermöglichen eine flexible Anycast-Funktionalität in *AIM*⁷. *AIM* Anycast Nachrichten werden von Routern jeweils auf dem Interface weitergeleitet, dessen *Distance Label* den kleinsten Wert aufweist. Der Anycast-Pfad ist also immer der kürzeste zum nächsten Gruppenmitglied (bei *Distance Label* vom „Typ 1“). Dabei ist zu beachten, dass bei *AIM* Anycastnachrichten immer entlang des Multicast Routingbaumes befördert werden, aber nicht den gleichen Weg wie Multicastnachrichten nehmen, die ja zum Beispiel bei CBT immer zuerst an den *Core Router* geschickt werden.

Beispiel (siehe 6.5.2): $R3$ sendet eine Anycastnachricht

Es ergibt sich der Pfad:

$$R3 \rightarrow 110 \rightarrow 11 \rightarrow 1 \rightarrow 10 \rightarrow R4$$

AIM liefert drei erweiterte Anycast Routingfunktionen: *Positional Anycast*, *Reverse Anycast* und *Reverse Positional Anycast*. Da nur der *Positional Anycast* in *RMA* verwendet wird, soll auf die beiden anderen hier nicht weiter eingegangen werden.

⁷In [Levine97] wurde diese Routingfunktion noch *Reachcasting* genannt, in [Levine99] wird dann die allgemein gebräuchliche Bezeichnung *Anycasting* verwendet.

Positional Anycast

Hier wird eine einzige Kopie einer Nachricht an den nächsten Empfänger (bezüglich der *Distance Labels*) in der Richtung auf ein bestimmtes Ziel hin zugestellt. Dieses Ziel wird mit dem *Positional Label* eines Routers angegeben. Der *Positional Anycast* ist in *AIM* einfach zu implementieren: Pakete, die auf dem Interface mit dem kleinsten *Distance Label* eintreffen, werden entsprechend dem *Positional Routing* in Richtung Ziel weitergeleitet, alle anderen werden über das Interface mit minimalem *Distance Label* weitergeleitet.

Beispiel (siehe 6.5.2): *R4* sendet eine *Positional Anycastnachricht* in Richtung Sender:

$$R4 \rightarrow 10 \rightarrow 1 \rightarrow 12 \rightarrow 120 \rightarrow \text{Sender}$$

zum Vergleich würde eine normale Anycastnachricht von *R4* folgenden Pfad nehmen:

$$R4 \rightarrow 10 \rightarrow 100 \rightarrow R1$$

6.5.3 Zuverlässiger Multicast mit AIM

Behandlung von NAKs

Empfänger, die den Verlust eines Pakets feststellen, senden ein NAK mittels *AIM Positional Anycast* in Richtung Sender. Das NAK erreicht so das nächste Gruppenmitglied in der Richtung zum Sender. Dies ist wichtig, da es wahrscheinlicher ist, dass Empfänger, die näher am Sender liegen, das fehlende Paket empfangen haben als Empfänger weiter unten im Multicastbaum. Ist das Paket beim Empfänger des NAK auch nicht vorhanden, so sendet er wieder ein NAK mittels *Positional Anycast* in Richtung Sender (falls er das nicht schon getan hat, doppelte NAKs werden unterdrückt).

Antwort auf NAKs

AIM bietet hier zwei Möglichkeiten. Ein einfacher Mechanismus bietet eine eins zu eins Übertragungswiederholung: Die fehlenden Pakete werden vom Empfänger des NAKs, der das Paket hat, über *Positional Routing* an den anfordernden Empfänger verschickt. Alternativ kann mit Hilfe des Masken-Mechanismus von *AIM* das fehlende Paket per *Subcast* an den ganzen Unterbaum des anfordernden Empfängers verschickt werden.

Senderbasiertes Verfahren

Optional bietet das *RMA Framework* auch ein senderbasiertes Verfahren mit *ACKs*.

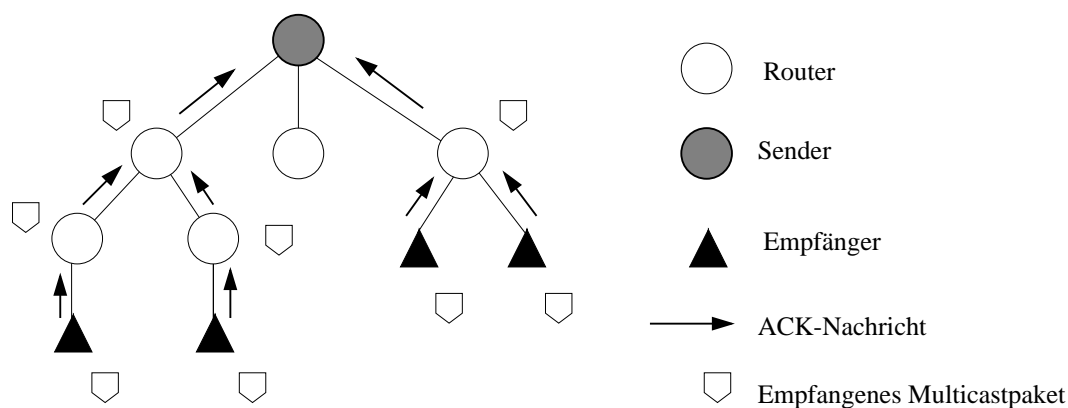


Abbildung 6.5.3: Strong Deletion: ACK wird erst gesendet, wenn alle Kinder ACK gesendet haben.

Aufbau des Kontrollbaums *RMA* definiert hierzu eine Eltern-Kind Beziehung zwischen den Mitgliedern der Multicastgruppe. Als die *Kinder* eines Mitgliedsknotens G werden die Knoten definiert, die G über einen *Positional Anycast* (in Richtung Sender) erreichen. Die Elternknoten müssen eine Liste ihrer aktuellen Kinderknoten unterhalten. Die Kinderknoten müssen ihre Eltern von ihrer Anwesenheit informieren:

- (Kinder-)Knoten senden eine *Register* Nachricht mit *Positional Anycast* (Richtung Sender).
- Elternknoten antworten mit einer *Register ACK* Nachricht, in der ihre Adresse enthalten ist.

Der entstehende Kontrollbaum entspricht somit genau dem Multicast-Routingbaum, was typisch für ein Router-basiertes Verfahren ist.

Bei Topologieänderungen, welche die Router durch Änderung der *Positional Labels* bemerken, wird eine Überprüfung der Eltern-Kind Beziehung nötig:

- (Kinder-)Knoten sendet eine *Parent-Query* Nachricht mit *Positional Anycast* (Richtung Sender).
- Der Elternknoten antwortet mit einer *Parent-Response* Nachricht, in der seine Adresse enthalten ist.
- Wenn die in der *Parent-Response* Nachricht enthaltene Adresse nicht mit der des bisherigen Elternknotens übereinstimmt, wird an diesen eine *Unregister* Nachricht geschickt. Es wird dann eine neue *Register* Nachricht verschickt (siehe oben).

Verlässt ein Mitglied die Gruppe, so wird an Eltern- und Kindknoten eine *Unregister* Nachricht verschickt.

Pufferfreigabe *RMA* bietet zwei Varianten zur Pufferfreigabe an.

Strong Deletion: Hier muss ein Elternknoten auf die ACKs aller Kindknoten warten, bevor er selbst ein ACK an seinen Elternknoten senden und den Speicher mit der entsprechenden Sequenznummer freigeben kann (siehe Abbildung 6.5.3).

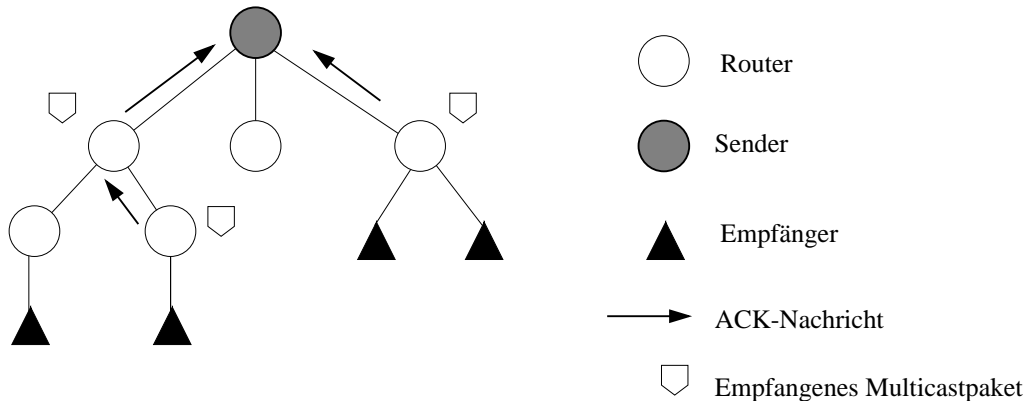


Abbildung 6.5.4: Weak Deletion: ACK wird gesendet, sobald Multicastpaket empfangen wurde.

Weak Deletion: Hier kann ein Elternknoten alle Pakete, die er selbst bekommen hat, sofort bestätigen, selbst wenn er noch nicht von allen Kindern eine Bestätigung erhalten hat (siehe Abbildung 6.5.4). Der Puffer kann aber auch erst freigegeben werden, wenn von allen Kindknoten ein ACK eingetroffen ist. Diese Variante braucht weniger Speicher als die erstere, hat aber das Problem, dass bei Topologieänderungen die Zuverlässigkeit nicht in jedem Fall garantiert ist.

6.6 Pragmatic General Multicast (PGM)

6.6.1 Überblick

PGM [Speakman99] wurde erstmals im Januar 1998 als *Internet Draft* veröffentlicht⁸. Der Routerhersteller *Cisco* ist an der Entwicklung von *PGM* beteiligt und bietet auch schon eine Implementierung in seinen Routern an. Wie bei den bisher besprochenen Protokollen verwendet auch *PGM* den Ansatz, den Multicastbaum der Netzwerkschicht als Baum für die Kontrollnachrichten zu verwenden. Auch ein Verfahren, wiederholte Pakete nur an den betroffenen Unterbaum auszuliefern (wie z.B. der *Subcast* bei *LMS*), ist enthalten. *PGM* enthält auch Mechanismen zur Flusskontrolle und zur Vorwärtsfehlerkorrektur, auf die hier aber nicht näher eingegangen werden soll. Wie die meisten bisher besprochenen Verfahren ist auch *PGM* empfangerbasiert, wodurch eine absolute Zuverlässigkeit nicht garantiert werden kann. Durch Bestätigungen für NAKs wird

⁸Im ersten Draft stand *PGM* noch für *Pretty Good Multicast*, was dann im zweiten Draft wegen rechtlicher Probleme durch *Pragmatic General Multicast* ersetzt wurde.

jedoch ein sehr hoher Grad an Zuverlässigkeit erreicht. Es wird auch garantiert, dass ein Empfänger den nicht reparierbaren Verlust eines Paketes entdeckt (so dass auf Anwendungsebene entsprechend reagiert werden kann).

PGM unterstützt mehrere Sender innerhalb der Multicastgruppe und macht keine Annahmen über das verwendete Multicast-Routingprotokoll.

6.6.2 Aufbau des Kontrollbaums

Wie oben schon erwähnt, wird auch hier der Multicast-Routingbaum als Kontrollbaum genutzt. Die Abbildung 6.6.1 zeigt einen Multicastbaum in *PGM*.

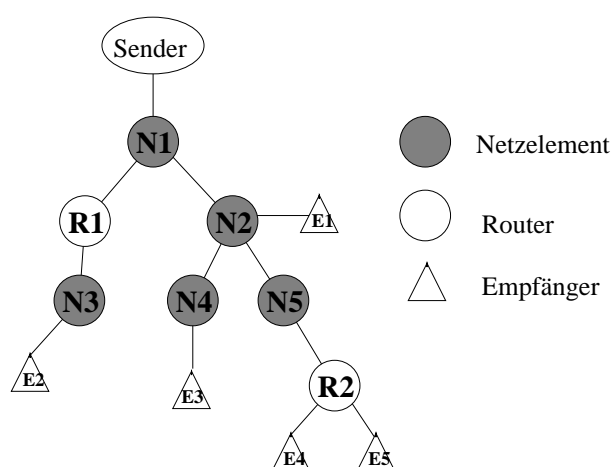


Abbildung 6.6.1: Elemente von PGM

Die *Netzelemente* sind Router, in denen die für *PGM* nötigen Erweiterung in der Netzwerkschicht vorhanden sind. Die in der Abbildung weiß dargestellten Router erlauben nur den normalen IP Multicast. Im optimalen Fall sind alle Router auch *PGM-Netzelemente*, das Protokoll sieht aber auch vor, dass Router ohne *PGM* Funktionen vorhanden sind, was jedoch die Effizienz des Protokolls beeinträchtigt.

Source Path Messages

In *PGM* verschickt ein Sender periodisch so genannte *Source Path Messages (SPM)*. Diese Nachrichten werden vom Sender als Multicast an die Gruppe verschickt. Normale IP-Router leiten *SPMs* wie jedes andere Multicastpaket weiter. *PGM Netzelemente* jedoch verarbeiten diese Pakete. Dies wird dadurch ermöglicht, dass der Sender *SPMs* mit der *IP Router Alert Option* [Katz97] verschickt. Auch andere Kontrollnachrichten werden in *PGM* mit dieser Option verschickt (mehr hierzu siehe unten). Die Aufgabe der *SPMs* ist, den *Netzelementen* und Empfängern jeweils das nächste *Netzelement* in Richtung des Senders bekanntzumachen. Die Nachrichten werden periodisch verschickt, um auf Änderungen in der Topologie reagieren zu können. Eine *SPM* beinhal-

tet die Adresse *SPM_PATH* des nächsten *Netzelements* in Richtung des Senders. Im Detail sieht das folgendermaßen aus (siehe Abbildung 6.6.1):

Sender: Der Sender setzt seine eigene (Unicast-)Adresse als *SPM_PATH* und verschickt die *SPM* per Multicast mit *Router Alert Option* (Beispiel: $Sender \rightarrow N1$).

Netzelement: Ein *Netzelement* extrahiert aus der empfangenen *SPM* die Information *SPM_PATH* und speichert diese zusätzlich zu den Routinginformationen für dieses $\langle \text{Gruppe}, \text{Sender} \rangle$ Paar (Beispiel: $N1$ speichert Unicast-Adresse des Senders). Das *Netzelement* versendet die *SPM* wie eine Multicastnachricht auf allen ausgehenden Interfaces für die Gruppe und setzt als *SPM_PATH* die Adresse des jeweiligen Interfaces ein (Beispiel: $N1 \rightarrow R1$ und $N1 \rightarrow N2$).

Router: Ein Router unterscheidet eine *SPM* nicht von einem normalen IP-Paket. Die *SPM* wird unverändert über Multicast weitergeleitet (Beispiel: $R1 \rightarrow N3$; $N3$ trägt in seine Routingtabelle $N1$ als *SPM_PATH* ein).

Empfänger: Wie *Netzelemente* extrahieren auch Empfänger den *SPM_PATH* und merken sich so die Adresse des nächsten *Netzelements* in Richtung Sender. Die *SPMs* werden vom Empfänger natürlich nicht weitergeleitet.

SPMs dienen auch der Flusskontrolle zwischen Sender und Empfänger, wie aber schon oben erwähnt, soll auf diesen Aspekt von *PGM* hier nicht eingegangen werden.

6.6.3 Fehlerkontrolle

Der Sender verwendet IP-Multicast, um Nachrichten an die Empfänger der Gruppe zu senden (im Beispiel 6.6.1 Empfänger $E1$ bis $E4$). Diese Nachrichten werden in der *PGM*-Terminologie als *ODATA* (engl.: original content data) bezeichnet. Die *Netzelemente* verhalten sich gegenüber *ODATA* wie die normalen Router und leiten sie nur entlang dem Routingbaum weiter. Empfänger entdecken den Verlust von *ODATA* an einer Lücke in den empfangenen Sequenznummern⁹ und erzeugen entsprechende selektive *NAKs*. Die *NAKs* werden vom Empfänger per Unicast an das nächste *Netzelement* in Richtung Sender (Adresse in *SPM_PATH*, siehe oben) geschickt. Im Gegensatz zu den bisher besprochenen *NAK*-basierten Verfahren werden hier *NAKs* bestätigt, was die Anfälligkeit des Protokolls gegen den Verlust von *NAKs* vermindert.

⁹oder durch die in *SPMs* enthaltene Flusskontrollinformation

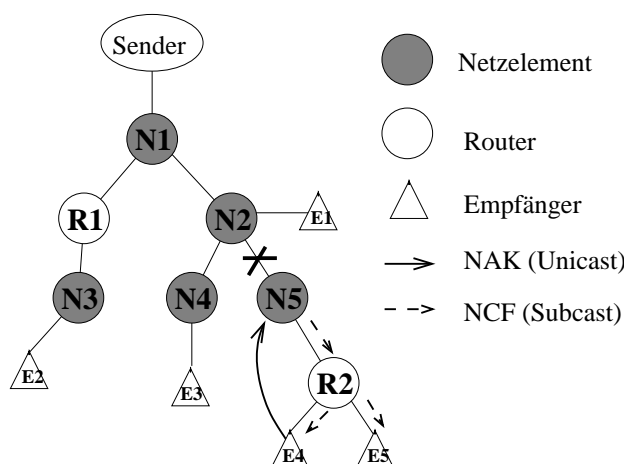


Abbildung 6.6.2: Empfänger sendet ein NAK

Behandlung von NAKs in Netzelementen:

Netzelemente bestätigen den Empfang eines *NAK* mit einem *NCF* (engl.: NAK confirmation). Das *NCF* wird auf dem Interface, auf dem das *NAK* empfangen wurde, über Multicast (mit gesetzter *Router Alert Option*) verschickt (siehe Abbildung 6.6.2). Es handelt sich hierbei um eine Art *Subcast* wie bei *OTERS* oder *LMS*. Zu beachten ist, dass das *Netzelement* nicht seine eigene Adresse, sondern die des Senders der entsprechenden *ODATA* als Absenderadresse für das *NCF* einträgt. Der Empfang eines *NAK* führt zu einem Eintrag in der *Fehlerkorrekturtable* des *Netzelements*. Es wird festgehalten, welche *NAKs* auf welchen Interfaces empfangen wurden. Diese Information wird später benötigt, um eine Übertragungswiederholung nur an die betroffenen Unterbäume auszuliefern. Wie beim Empfänger wird auch im *Netzelement* ein *NAK* über Unicast an das nächste *Netzelement* in Richtung Sender (Adresse aus *SPM_PATH*) weitergeleitet.

Behandlung von NAKs beim Sender:

Der Sender beantwortet ein *NAK* sofort mit einem *NCF*, das über Multicast und *Router Alert Option* an die ganze Gruppe verschickt wird (mehr über *NCFs* und deren Wirkung weiter unten). Dann wird die entsprechende Übertragungswiederholung durchgeführt. In der *PGM*-Terminologie wird ein wiederholtes (*ODATA*-)Paket als *RDATA* (engl.: retransmission data) bezeichnet. Im Gegensatz zu *ODATA* Paketen, die über normalen Multicast gesendet werden, wird *RDATA* mit Multicast und gesetzter *Router Alert Option* verschickt.

Behandlung von RDATA in Netzelementen:

Wegen der gesetzten *Router Alert Option* wird *RDATA* von jedem *Netzelement* bearbeitet. Findet ein *Netzelement* keinen entsprechenden Eintrag in seiner Fehlerkor-

rekturtabelle (hat kein entsprechendes *NAK* oder *NCF* empfangen), so wird *RDATA* verworfen. Findet sich ein entsprechender Eintrag, so wird *RDATA* auf allen Interfaces der Liste weitergeleitet (der Header des *RDATA* Pakets bleibt unverändert) und der entsprechende Eintrag in der Fehlerkorrekturtabelle gelöscht (siehe Abbildung 6.6.3).

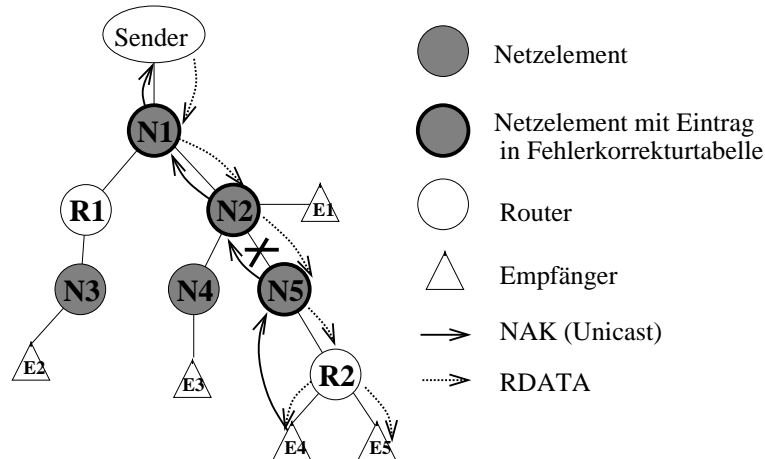
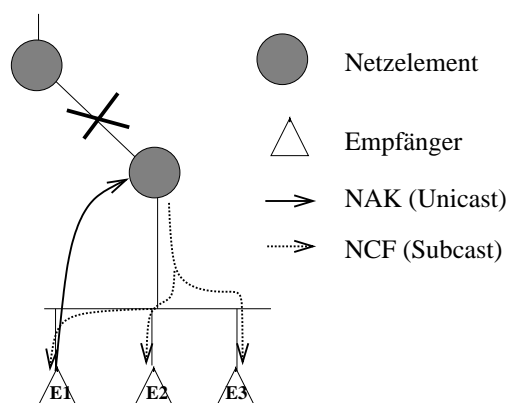


Abbildung 6.6.3: Behandlung von *RDATA*

Timeout Intervalle bei *NAKs*:

Bisher war die Behandlung von *NAKs* in *PGM* zum besseren Verständnis noch vereinfacht dargestellt: Beim Erzeugen eines *NAK* beachtet ein Empfänger vier Timeoutintervalle:

1. Das erste Intervall wird als *NAK_RB_IVL* (engl.: *NAK random back-off interval*) bezeichnet. Während dieses Intervalls wird das Versenden eines *NAK* abhängig von einer Zufallszahl verzögert, was verhindert, dass viele Empfänger gleichzeitig für gleiche *ODATA NAKs* versenden. Erreicht den Empfänger während dieses Intervalls ein zu *ODATA* gehöriges *NCF*, so wird kein eigenes *NAK* gesendet (Abbildung 6.6.4: *E2* und *E3* empfangen während *NAK_RB_IVL* ein *NCF* und versenden kein *NAK*). Wenn der Timer abläuft, wird wie oben beschrieben ein *NAK* verschickt.
2. Während des *NAK_RPT_IVL* (engl.: *NAK repeat interval*) wiederholt der Empfänger sein *NAK*, bis er ein zugehöriges *NCF* empfängt. Empfängt er dies bis zum Ende des Intervalls nicht, so liegt ein nicht behebbarer Datenverlust vor.
3. Während *NAK_RDATA_IVL* (engl.: *NAK RDATA interval*) wartet der Empfänger auf *RDATA*. Wird während des Intervalls *RDATA* nicht empfangen, so wird erneut ein *NAK* erzeugt (d.h. bei Punkt 1 begonnen).
4. Schließlich bestimmt *NAK_GEN_IVL* wie lange ein Empfänger erneute *NAKs* verschickt, während er auf *RDATA* wartet. Läuft das Intervall ab, bevor *RDATA* empfangen wird, so bedeutet auch dies unwiederbringlichen Datenverlust.

Abbildung 6.6.4: Vermeidung von *NAKs*

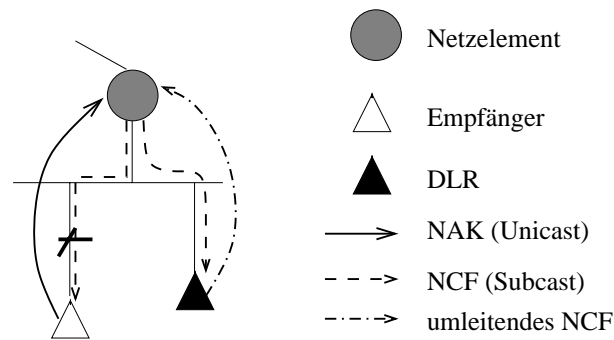
Bei den *Netzelementen* wird bezüglich der Weiterleitung der *NAKs* ziemlich ähnlich wie bei den Empfängern verfahren. Allerdings gibt es die folgenden wichtigen Unterschiede:

1. Es gibt keine Verzögerung vor der Weiterleitung (kein *NAK_BO_IVL*), da sich die Verzögerungen sonst mit jedem Hop aufsummieren würden.
2. Wenn auf ein bestätigtes *NAK* hin das zugehörige *RDATA* nicht empfangen wird, so wird kein weiteres *NAK* verschickt, sondern der entsprechende Eintrag in der Fehlerkorrekturtafel gelöscht und darauf vertraut, dass ein Empfänger erneut ein *NAK* verschickt.
3. In *Netzelementen* wird die Erzeugung eines *NAK* nicht durch die Ankunft der entsprechenden *ODATA* unterdrückt, da *ODATA* als normaler Multicast (ohne *Router Alert Option*) verschickt und deshalb gar nicht „bemerkt“ wird.

6.6.4 Optionales Element in *PGM*: *DLRs*

PGM bietet die Möglichkeit, dass *RDATA* nicht nur vom Sender, sondern auch von so genannten *DLRs* (engl.: designated local repairer) gesendet wird. Dies wird dadurch erreicht, dass *DLRs* auf normale *NCFs* mit einem speziellen (Umleitungs-) *NCF* antworten, in dem sie ihre eigene Adresse als „Alternative“ zum Sender der *ODATA* verbreiten. Empfänger dieses umleitenden *NCF* können dann ihre nächsten *NAKs* an den *DLR* senden. *DLRs* verhalten sich beim Empfang von *NAKs* dann wie ein Sender (siehe Abbildung 6.6.5).

Empfängt ein *DLR* solch ein umgeleitetes *NAK* und steht die entsprechende *ODATA* zur Verfügung, so schickt er ein so genanntes *NULL NAK* an den Sender. Dies dient nur der Information des Senders (Flußkontrolle) und verursacht keine Übertragungswiederholung.

Abbildung 6.6.5: *DLR* antwortet mit umleitendem *NCF*

6.7 Weitere Verfahren

In der Literatur sind noch weitere Router-basierte Verfahren vorgeschlagen worden, die der Vollständigkeit halber hier kurz erwähnt werden sollen. Das in [Levine98] beschriebene *Tracer* verwendet eine Kombination aus ERS und *Mtrace*¹⁰ um einen am Auslieferungsbaum orientierten Kontrollbaum aufzubauen. In *Express* [Holbrook99] wird die Vermittlungsschicht um einen Zählmechanismus erweitert. Es werden für jedes ausgehende Interface eines Routers die Anzahl der darüber erreichten Gruppenmitglieder gezählt. Dies kann verwendet werden, um zu bestimmen, ob alle ACKs eingetroffen sind, ohne die Empfänger explizit zu kennen. *Search Party* [Costello99] führt einen so genannten *Randomcast* ein. Der bei Search Party verwendete Randomcast ist eine Anycast (siehe auch 2.2.4), wobei der Empfänger ein (nach einer vorgegebenen Wahrscheinlichkeitsverteilung) zufällig ausgewähltes Gruppenmitglied ist. Sonst wird hier ein an LMS¹¹ angelehntes Protokoll verwendet. Der Randomcast soll verhindern, dass wenige Empfänger durch NAKs und Übertragungswiederholungen zu stark belastet werden.

6.8 Prinzipien der Router-basierten Ansätze

Hier soll noch einmal ein Überblick über die Features der Router-basierten Verfahren gegeben werden. Tabelle 6.8.1 zeigt, welche Features von welchen Verfahren eingesetzt werden.

¹⁰Siehe 6.3.2.

¹¹Siehe 6.2.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
LMS	X		X	X			X		
OTERS				X		X			
RMA	X	X		X	X		X		
ARM	X	X		X				X	X
PGM		X	X	X		X	X	X	

Abbildung 6.8.1: Features der vorgestellten Router-basierten Verfahren

- (a) Unterscheidung zwischen *Upstream* und *Downstream*.
- (b) Besondere Behandlung von Kontrollnachrichten und Übertragungswiederholungen.
- (c) Verwendet zusätzliche Kontrollnachrichten.
- (d) Verwendet *Subcasting*.
- (e) Verwendet *Anycasting*.
- (f) Benutzt *Multicast Route Backtracing*.
- (g) Router speichert zusätzliche Statusinformationen pro Sender und Gruppe.
- (h) Router speichert zusätzliche Statusinformationen zu Kontrollnachrichten.
- (i) Router speicher Datenpaket im Cache.

Kapitel 7

Vergleich ausgewählter Protokolle

In diesem Abschnitt geht es um den Vergleich einiger ausgewählter Protokolle für zuverlässige Gruppenkommunikation. Es wird auf die Vergleichskriterien, die Implementierung des Simulationscodes und die Durchführung der Simulationen eingegangen.

Inhaltsangabe

7.1	Überblick	68
7.2	Vergleichskriterien	68
7.3	Realisierung der Simulationen	73

7.1 Überblick

Als Protokolle für die Simulationen wurden *OTERS* (siehe 6.3) und *PGM* (siehe 6.6) von den Router-basierten Protokollen sowie *ERS* (siehe 5.2.1), *ERA* (siehe 5.2.2) und *TRS* (siehe 5.2.3) der nicht Router-basierten Verfahren ausgewählt. Die genannten nicht Router-basierten Verfahren sind natürlich nur Verfahren zum Aufbau des Kontrollbaums. Als erstes Vergleichskriterium wurde daher der Nachrichtenaufwand für den Aufbau des Kontrollbaums beziehungsweise den entsprechenden Aufbau der Statusinformationen bei den Router-basierten Verfahren gewählt.

Um die aufgebauten Kontrollbäume unter einheitlichen Bedingungen vergleichen zu können, wurde TMTP (siehe 5.3.3) so abgeändert, dass es mit den jeweiligen Protokollen aufgebaute Kontrollbäume verwendet. Da aber, wie oben schon erwähnt, die Router-basierten Verfahren keine expliziten Kontrollbäume aufbauen, konnten für diese Messungen nur die nicht Router-basierten Verfahren herangezogen werden. Mit TMTP wurde dann der Durchsatz bei den verschiedenen Bäumen ermittelt.

7.2 Vergleichskriterien

7.2.1 Aufwand Baumaufbau

Hier soll, wie oben schon erwähnt, der Nachrichtenaufwand für den Aufbau des Kontrollbaums beziehungsweise den entsprechenden Aufbau der Statusinformationen gemessen werden. Hierzu tritt eine bestimmte Anzahl von Empfängern der Multicastgruppe bei. Die hierdurch anfallenden Nachrichten werden gezählt.

Grundsätzliche Überlegungen

Da *OTERS*, *PGM* und *ERA* periodische (limitierte) Multicastnachrichten verwenden, ist zu erwarten, dass sie vom Nachrichtenaufwand *ERS* und *TRS* unterlegen sind, da hier die Eingliederung rein *kindbasiert* vor sich geht. Da ein *Shared Tree* Routingprotokoll für *ERS* und *ERA* sehr ungünstig wäre, wurde der Vergleich fairerweise nur mit einem *Source Routed Tree* Routingprotokoll durchgeführt.

Bestimmung des Nachrichtenaufwands

Als Maßzahlen für den durch den Aufbau von Kontrollbaum und Statusinformationen verursachten Nachrichtenaufwand wurde folgende Zählweise festgelegt (Routingprotollnachrichten, wie zum Beispiel *Prune* oder *Graft* Nachrichten, werden nicht gezählt):

Gesendete Nachrichten: Jedes Sendeereignis wird gezählt, unabhängig davon, ob es sich um eine Unicast- oder eine Multicastnachricht handelt. Im Beispiel (Abbildung 7.2.1) wäre dies ein Sendeereignis (gesendet von Knoten 9).

Empfangene Nachrichten: Hier werden Empfangsereignisse gezählt. Wird ein bestimmtes Paket an einem Knoten mehrfach empfangen (das entspricht mehreren Empfängern in einem lokalen Netz) so wird das Ereignis nur einmal gezählt. Im Beispiel wären das vier Empfangsereignisse an den Knoten 6, 8, 10 und 12. Man beachte, dass trotz der zwei Empfänger an Knoten 10 nur ein Empfangsereignis gezählt wird.

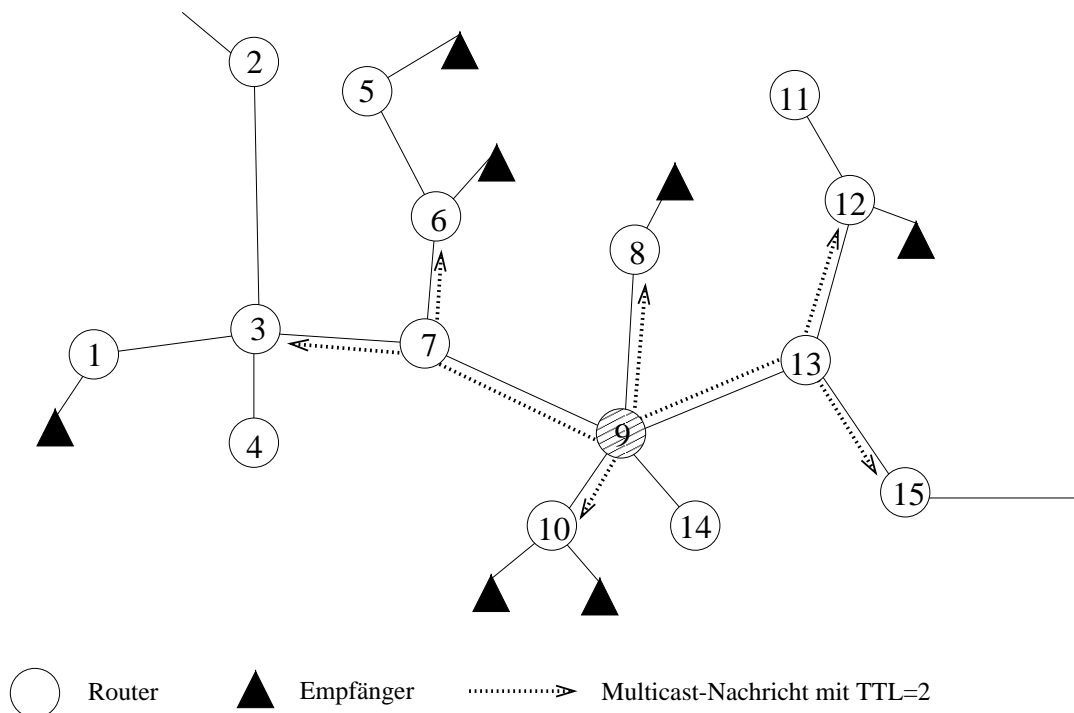


Abbildung 7.2.1: Beispiel für Paketstatistik

Packet Hops: Hier wird für alle Paket die Anzahl der passierten Hops aufsummiert. Dies ist wohl das beste Maß, da hier die Belastung des Netzwerks am besten gemessen wird. Im Beispiel (Abbildung 7.2.1) würden neun *Packet Hops* gezählt.

Vergleichsszenario

Ermittelt werden die oben erläuterten Größen für Netzwerke verschiedener Größe (bezüglich der Anzahl der Knoten) und mit Gruppen unterschiedlicher Größe. Die Gruppenmitglieder sind dabei zufällig im Netzwerk verteilt und treten zufällig über einen bestimmten Zeitraum verteilt der Multicastgruppe bei.

7.2.2 Durchsatz

Als Maß für die Qualität der zu vergleichenden Kontrollbäume wird der erzielte Durchsatz mit TMTP gemessen. Hierbei wird ein modifiziertes TMTP-Protokoll verwendet, das einen fertigen Kontrollbaum übergeben bekommt. Das eingesetzte Verfahren verwendet, wie in 5.3.3 beschrieben, NAKs und aggregierte ACKs. Der Sender sendet mit einer festgelegten maximalen Datenrate, die durch einen Fenstermechanismus begrenzt wird. Der Durchsatz wird für verschiedene Szenarien gemessen, unter anderem auch bei Paketverlust (mehr siehe unten).

Grundsätzliche Überlegungen

Der erzielte Durchsatz hängt durch den verwendeten Fenstermechanismus (wenn die Fenstergröße nicht sehr groß ist) hauptsächlich davon ab, wieviel Zeit zwischen dem Senden des Multicastpakets und der Ankunft des zugehörigen aggregierten ACKs verstreichen¹. Hierfür ist zum einen die Höhe der Bäume, zum anderen die Verzögerung zwischen den Knoten des Kontrollbaums maßgeblich.

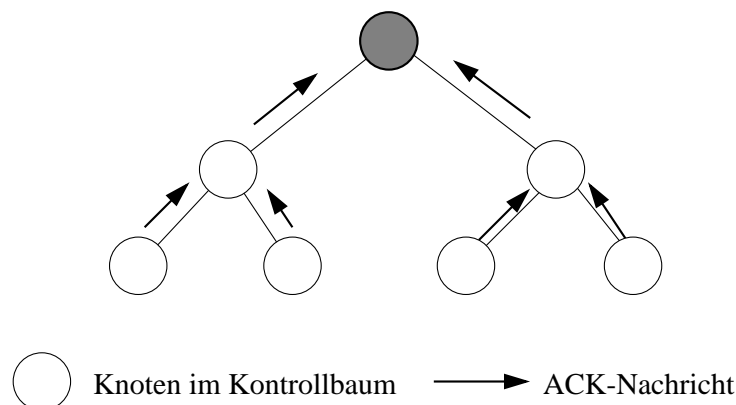


Abbildung 7.2.2: Beispiel für einen guten Kontrollbaum

Die Abbildungen 7.2.2 und 7.2.3 zeigen anschaulich, dass die Gesamtverzögerung hauptsächlich vom längsten Pfad im Kontrollbaum abhängt und daher ein Kontrollbaum mit minimaler Höhe angestrebt werden sollte, was ja die Strategie bei TRS-MH² ist.

Wie schon erwähnt, ist ein weiterer Aspekt die Verzögerung zwischen den Knoten des Kontrollbaums. Da ja bei den hier betrachteten Ansätzen der Kontrollbaum in der Regel nicht dem Routingbaum entspricht, können die Knoten des Kontrollbaums relativ weit auseinander liegen.

Abbildung 7.2.4 zeigt die Knoten eines Kontrollbaums in einem Netzwerk. In Abbildung 7.2.5 sieht man dann einen möglichen Kontrollbaum. An den Kanten ist der Abstand der Knoten im Netzwerk eingezeichnet: Als Maß ist hier die Anzahl der Hops

¹Diese Zeit wird oft als *Round Trip Delay* bezeichnet.

²siehe 5.2.3

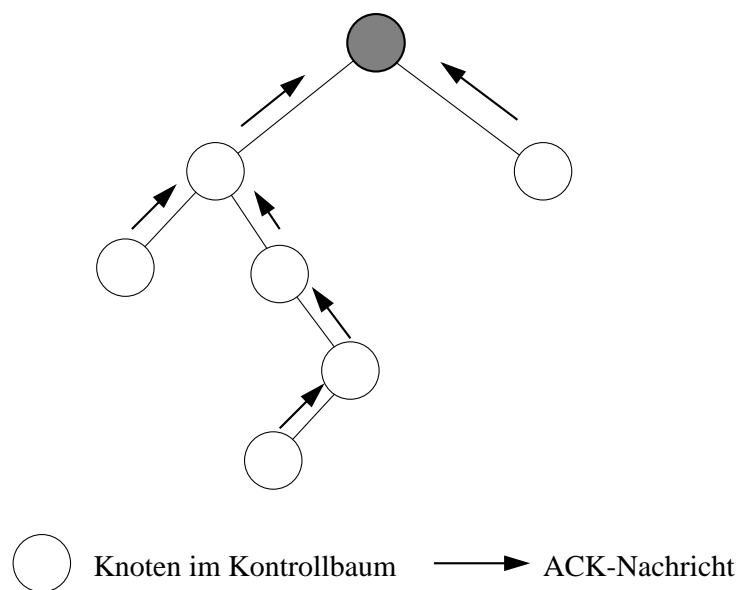


Abbildung 7.2.3: Beispiel für entarteten Kontrollbaum

gewählt. Ein besseres Maß wäre noch die Verzögerung zwischen beiden Knoten. Das Beispiel zeigt schon deutlich, dass eine Optimierung auf minimale Höhe *und* minimale Verzögerung keineswegs trivial ist. Unter dem Aspekt der Minimierung der Entfernungen ist *ERS* ein sehr geeignetes Protokoll, da es eben versucht, die Hops zwischen zwei Knoten des Kontrollbaums zu minimieren. Allerdings muss man einschränkend sagen, dass dies nur dann gut funktioniert, wenn alle Verbindungen im Netzwerk ungefähr die gleich Verzögerung haben und ein *Source Routed Tree* Routing Protokoll verwendet wird.

Ein weiterer Faktor, der den Durchsatz mitbestimmt, ist die Verlustrate. Hiermit ist die Wahrscheinlichkeit gemeint, mit der Datenpakete verloren gehen. Ein verlorenes Paket verlängert die Zeit, bis ein aggregiertes ACK empfangen werden kann. Ein trivialer Zusammenhang ist, dass der Durchsatz mit zunehmender Verlustrate sinkt. Allerdings hat auch hier der Aufbau des Kontrollbaums einen Einfluss darauf, wie stark sich ein Paketverlust auswirkt: Da bei *TMTF* die inneren Knoten des Kontrollbaums die Funktion eines *Domain Manager (DM)*³ haben, ist hier die Verzögerung zum direkten Vorgängerknoten ein wichtiger Faktor, wie schnell eine (durch NAK veranlasste) Übertragungswiederholung stattfinden kann.

Bestimmung des Durchsatzes

Die Flusssteuerung beruht auf einer Kombination aus maximaler Senderate und einem Fenstermechanismus. Die Senderate gibt an, wieviele Pakete pro Zeiteinheit gesendet werden sollen. Bei jeder Sendeoperation wird geprüft, ob der aktuelle Zustand des Fensters das Senden eines weiteren Pakets zulässt: Ist die Prüfung erfolgreich, wird

³siehe 5.3.3

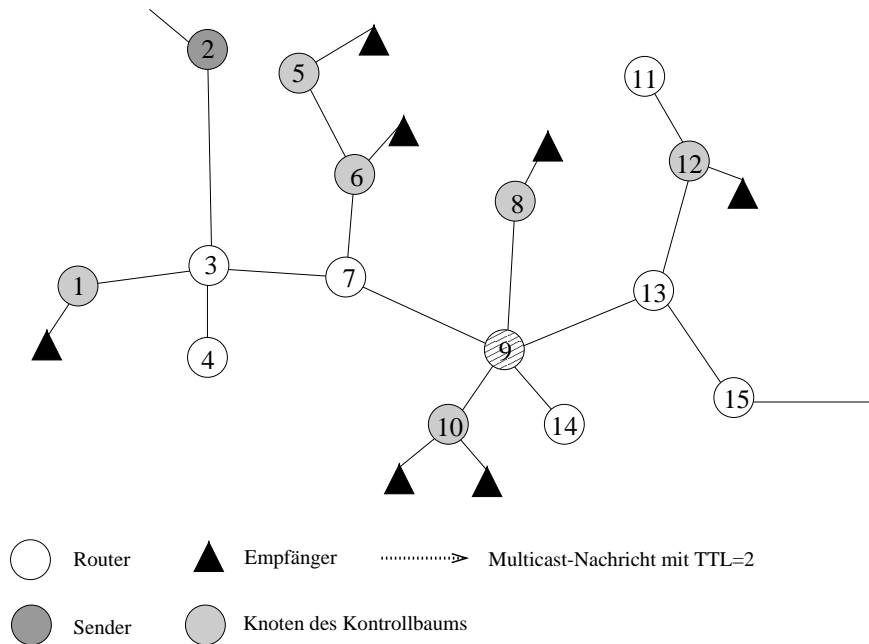


Abbildung 7.2.4: Knoten des Kontrollbaums in einem Netzwerk

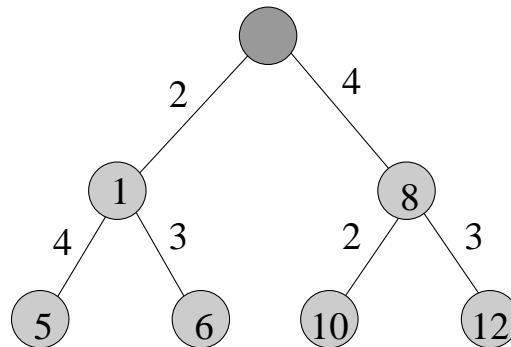


Abbildung 7.2.5: Ein möglicher Kontrollbaum zu 7.2.4

das Paket gesendet, falls nicht, wird nichts gesendet und nur der Zähler der erfolglosen Sendeoperation erhöht. Der ermittelte Durchsatz ergibt sich dann als das Verhältnis zwischen erfolgreichen Sendeversuchen und allen Sendeversuchen (siehe Abbildung 7.2.6).

Vergleichsszenario

Für die Messung des Durchsatzes wurde folgendes Szenario gewählt: Alle Empfänger sind der Multicastgruppe schon beigetreten und der Kontrollbaum ist schon aufgebaut. Der Sender sendet dann über einen festgelegten Zeitraum Daten an die Multicastgruppe. Der Durchsatz wird dabei nach dem beschriebenen Algorithmus bestimmt. Dies wird für Netzwerke verschiedener Größen, verschiedene Gruppengrößen, mit und ohne Paketverlust durchgeführt. Für den Paketverlust wird folgendes Fehlermodell benutzt: Jedes Datenpaket wird von jedem Netzwerkknoten, den es passiert, mit einer

Algorithmus: Bestimmung Durchsatz

```
Sendeversuche := 0;
Erfolgreiche_Senderversuche := 0;
Datenintervall := 1 / Maximale_Datenrate
while (Daten zu senden) do
  Sendeversuche := Sendeversuche + 1;
  if (Sendefenster ist offen)
  then
    sende Datenpaket;
    Erfolgreiche_Senderversuche := Erfolgreiche_Senderversuche + 1;
  end
  sleep(Datenintervall);
end
Durchsatz := Erfolgreiche_Senderversuche / Sendeversuche;
```

Abbildung 7.2.6: Algorithmus: Bestimmung Durchsatz

bestimmten Wahrscheinlichkeit verworfen. Damit ist die Wahrscheinlichkeit, dass ein bestimmtes Paket verloren geht, abhängig von der gewählten Verlustrate und der Zahl der passierten Hops.

7.3 Realisierung der Simulationen

7.3.1 NS

Die Simulationen wurden mit dem UCB/LBNL/VINT Netzwerk Simulator (ns-2) [NS2] durchgeführt. Der Simulator erlaubt dem Benutzer, das Verhalten von Netzwerken und Protokollen zu simulieren. Es können beliebige Netzwerktopologien verwendet werden, die aus Knoten und Netzwerkverbindungen⁴ bestehen. Die verwendeten Netzwerktopologien wurden mit dem GT-ITM Topologiegenerator [GT-ITM] erzeugt. An die Knoten können Protokollinstanzen, so genannte *Agenten*, gebunden werden. Verschiedene Multicast-Routingprotokolle werden unterstützt. Für alle durchgeführten Simulationen wurde DVMRP (siehe 3.3.2) verwendet.

Der Simulator ist in den Sprachen *C++* und *Tcl* implementiert. Objekte können über beide Sprachen angesprochen werden. Grundsätzlich ist die Arbeitsweise so, dass Funktionen, die auf jedes Datenpaket angewendet werden, in *C++* implementiert sind und für die Konfigurationsebene *Tcl* verwendet wird.

Zu beachten ist, dass beim Senden von Multicastnachrichten in großen Netzen die Laufzeit der Simulationen und der benötigte Speicherbedarf sehr stark ansteigen. Die simulierten Netzwerktopologien wurden deshalb auf maximal 1750 Knoten beschränkt.

⁴engl.: *Links*

7.3.2 Simulationsumgebung

Als Umgebung war eine Installation des NS in der Version 2.1b3 mit einer Implementierung der Verfahren ERS, ERA, TRS-PS, TRS-RC und TRS-MH vorhanden. Für OTERS, TMTP und PGM war Simulationscode für ältere Versionen von NS vorhanden, der andere Netzwerktopologien benutzte und verschiedene Tcl-Konfigurationsskripte verwendete. Auch wurden unterschiedliche Mess- und Auswertungsmethoden verwendet. Es ergab sich also als erste Aufgabe, eine einheitliche Simulationsumgebung für die Protokolle herzustellen. Hierzu wurde die zur Zeit aktuelle Version 2.1b5 von NS installiert und der Simulationscode der genannten Verfahren dorthin portiert. Hierzu war unter anderem die Anpassung einiger Schnittstellen und einige Anpassungen im Routing des Simulators nötig.

7.3.3 Messung Baufaufbau

Integration der Protokolle

Für die Simulationen zum Baufaufbau waren so genannte *Lastfiles* vorgegeben. Abbildung 7.3.1 zeigt einen Ausschnitt aus einem Lastfile. In der ersten Spalte steht die Simulationszeit, die zweite gibt eine Knotennummer an, die dritte Spalte enthält einen Befehl, die vierte Spalte enthält die Multicastgruppe. Die letzte Spalte enthält einen optionalen Parameter. Die erste Zeile bedeutet, dass zum Zeitpunkt 0 der Knoten 166 eine neue Multicastgruppe mit der Adresse 1073741824 erzeugt. Zum Zeitpunkt 6 wird dann am Knoten 42 der Beitritt zu dieser Multicastgruppe veranlasst (usw.).

0	166	CreateGroup	1073741824
6	42	JoinGroup	1073741824
8	128	JoinGroup	1073741824
13	114	JoinGroup	1073741824

Abbildung 7.3.1: Ausschnitt aus einem Lastfile

Eine Aufgabe im Rahmen dieser Arbeit war, diesen Mechanismus bei allen simulierten Protokollen anzuwenden.

Implementierung der Paketstatistikfunktion

Eine Frage bei der Erhebung des Nachrichtenaufwands (wie in 7.2.1 definiert) war, wie die entsprechenden Funktionen in das Simulationssystem zu integrieren sind. Ein oft verwendetes Verfahren ist, dass in den jeweiligen Protokollagenten Zähler implementiert werden, die bei Sende- und Empfangsereignissen erhöht werden. Da dies aber dann für jedes neue Protokoll gemacht werden muss, ist dieser Ansatz für den Vergleich vieler verschiedener Protokolle nicht geeignet. Ein weiteres Problem ist, dass auf diese Weise das wichtige Maß der *Packet Hops* nicht ermittelt werden kann.

Aus diesem Grund wurde die vom Simulator gebotene Möglichkeit zum Erzeugen so genannter *Tracefiles*⁵ genutzt. In diesem Tracefile werden so genannte *Paketereignisse* aufgezeichnet. Abbildung 7.3.2 zeigt zur Veranschaulichung einen Auszug aus einem Tracefile.

```

.
.
+ 1.84566 0 2 PGM 1000 ----- 2 0.1 3.2 102 611
- 1.84566 0 2 PGM 1000 ----- 2 0.1 3.2 102 611
r 1.84609 0 2 cbr 210 ----- 0 0.0 3.1 225 610
+ 1.84609 2 3 cbr 210 ----- 0 0.0 3.1 225 610
d 1.84609 2 3 cbr 210 ----- 0 0.0 3.1 225 610
- 1.8461 2 3 cbr 210 ----- 0 0.0 3.1 192 511
.
.
.

```

Abbildung 7.3.2: Ausschnitt aus einem Tracefile

Die erste Spalte zeigt den Typ des Ereignisses, wobei „+“ für das Einreihen eines Pakets in eine Warteschlange und „-“ das Verlassen einer Warteschlange anzeigt. Der Typ „r“ zeigt die Ankunft eines Pakets an einem Knoten an. Typ „d“ bedeutet, dass ein Paket verworfen wurde (zum Beispiel wegen eines Pufferüberlaufs). In der zweiten Spalte befindet sich die simulierte Zeit. Die nächsten beiden Spalten enthalten die Nummern der beiden Knoten, zwischen denen das Paket ausgetauscht wird. Die fünfte Spalte enthält dann eine Bezeichnung für den Pakettyp. Für diese Betrachtung ist dann noch die letzte Spalte interessant, die eine eindeutige Sequenznummer für das Paket enthält⁶.

Zur Auswertung dieser Tracefiles wurde eine Routine geschrieben, die daraus wie oben beschrieben die gesendeten Nachrichten und die *Packet Hops* ermittelt. Für die Bestimmung der empfangenen Pakete war eine Änderung am Simulator nötig, da sich die Empfangereignisse an einem oder mehreren mit einem Knoten verbundenen Empfängern mit den vorhandenen Daten des Tracefiles nicht ermitteln lassen. Dazu wurde „R“ als weiteres Ereignis eingeführt. Das Ereignis „R“ zeigt dann den Empfang eines Pakets durch einen mit einem Knoten verbundenen Agenten an.

7.3.4 Messung Durchsatz

Ausgangspunkt war eine für eine ältere Version des Simulators vorhandene Implementierung von TMTP. Die Implementierung beinhaltete den Aufbau des Kontrollbaums mit ERS, für den zuverlässigen Multicast jedoch nur das empfängerbasierte Verfahren.

⁵von engl.: *to trace*: aufzeichnet, verfolgen, nachspüren

⁶Diese Sequenznummer ist für die interne Nutzung im Simulator gedacht und ist nicht zu verwechseln mit Sequenznummern, die in den zu simulierenden Protokollen verwendet werden.

Die Aufgabe bestand also darin, die Implementierung um eine Flusststeuerung mit aggregierten ACKs zu erweitern und eine Möglichkeit zu schaffen, den Baumaufbau auch mit anderen Verfahren durchzuführen.

Schnittstelle zur Übergabe des Kontrollbaums

Als Form der Übergabe wurde eine Tabelle⁷ verwendet, die alle Knoten des Kontrollbaums enthält und zu jedem Knoten den Vorgängerknoten angibt.

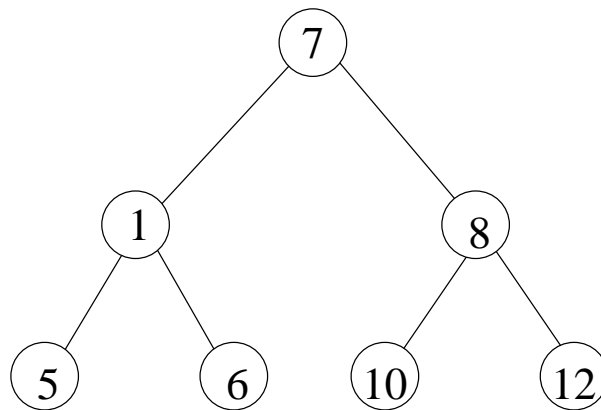


Abbildung 7.3.3: Beispiel für einen zu übergebenden Kontrollbaum

Abbildung 7.3.3 zeigt ein Beispiel für einen Kontrollbaum. Abbildung 7.3.4 zeigt die Tabelle, die dann an TMTP übergeben wird. Bei der Wurzel wird als Vorgängerknoten -1 angegeben.

7	-1
1	7
5	1
6	1
10	8
12	8
8	7

Abbildung 7.3.4: Tabelle zum Kontrollbaum in 7.3.3

Diese Tabelle wird dann verwendet, um den entsprechenden Kontrollbaum bei TMTP aufzubauen. Da TMTP (und auch die Implementierung im Simulator) keine zentrale Datenstruktur verwendet, sondern die Statusinformationen in den jeweiligen Protokollagenten gespeichert werden, wurden zwei neue Nachrichtentypen zum Aufbau des Kontrollbaums eingeführt:

⁷Als *Tel Array* realisiert

FORCE_JOIN_DR: Erlaubt es einem Knoten, sich an seinen in der Tabelle angegebenen Vorgängerknoten anzubinden (ohne die sonst nötigen Such- und Antwortnachrichten).

FORCE_SOURCE: Eine *ANNOUNCE_SOURCE* Nachricht dient bei TMTP unter anderem dazu, allen Mitgliedern der Multicastgruppe die Adresse des Senders mitzuteilen. Dadurch wird dann eine *Expanding Ring Search* ausgelöst (siehe auch 5.2.1). Die *FORCE_SOURCE* Nachricht hat die gleichen Funktionen, löst aber keine *Expanding Ring Search* aus.

Flusssteuerung in TMTP

Da wie oben erwähnt nur der empfangenerinitiierte Teil von TMTP implementiert war, mussten die Protokollagenten so ergänzt werden, dass jeder Knoten des Kontrollbaums seine direkten Nachfolger kennt. Wie in 4.6 beschrieben, wird ein ACK gesendet, sobald alle ACKs der Nachfolgerknoten eingetroffen sind.

Beim Sender wurde eine Kombination von Fenstermechanismus und maximaler Senderate verwendet. Bei jedem Sendeereignis wird das Fenster um einen Speicherplatz verkleinert, und beim Empfang eines aggregierten ACKs wird der Speicher wieder freigegeben.

Die maximale Senderate wurde durch einen *Timer* realisiert. Jedesmal, wenn er abläuft, ruft der Sender seine Sendemethode auf: Gesendet wird aber nur, wenn es der Zustand des Fensters zulässt (siehe 7.2.2).

Simulation von Paketverlusten

Zum Vergleich der Kontrollbäume der untersuchten Verfahren sollten auch Durchsatzmessungen bei Paketverlusten durchgeführt werden. Eine häufig verwendete Methode zur Simulation von Paketverlusten ist die Verwendung einer so genannten *Hintergrundlast*. Diese kann zum Beispiel durch zufällig im Netzwerk verteilte Sender und Empfänger erzeugt werden, die Daten austauschen und das Netzwerk so belasten, dass es manchmal zu Pufferüberläufen und dadurch zu Paketverlusten kommt. Wenn die Hintergrundlast geeignet gewählt wird, kommt diese Methoden den Bedingungen in tatsächlichen Netzwerken wie dem Internet am nächsten. Ein Problem bei der Verwendung von Hintergrundlast ist jedoch, dass sie in der Simulation sehr viel Rechenzeit und Speicherplatz benötigt. Außerdem ist es schwer, die Hintergrundlast so zu gestalten, dass genau die beabsichtigten Verlustraten auftreten. Aus diesen Gründen wurde hier ein anderer Ansatz gewählt: Wie schon in 7.2.2 geschildert, wird der Datenverlust so implementiert, dass ein Paket von jedem Netzwerkknoten, den es passiert, mit einer bestimmten Wahrscheinlichkeit verworfen wird. Diese Wahrscheinlichkeit lässt sich dann für jede Simulation geeignet einstellen.

Im folgenden soll kurz erläutert werden, wie dieser Mechanismus in den Simulator integriert wurde. Abbildung 7.3.5 zeigt zwei Knoten einer Simulation und die innere

Struktur eines Knotens. Wichtig sind in diesem Zusammenhang nur die so genannten *Classifier*-Objekte `classifier_` und `multiclassifier_`, wobei das erstere für das Unicast-Routing und das letztere für das Multicast-Routing zuständig ist.

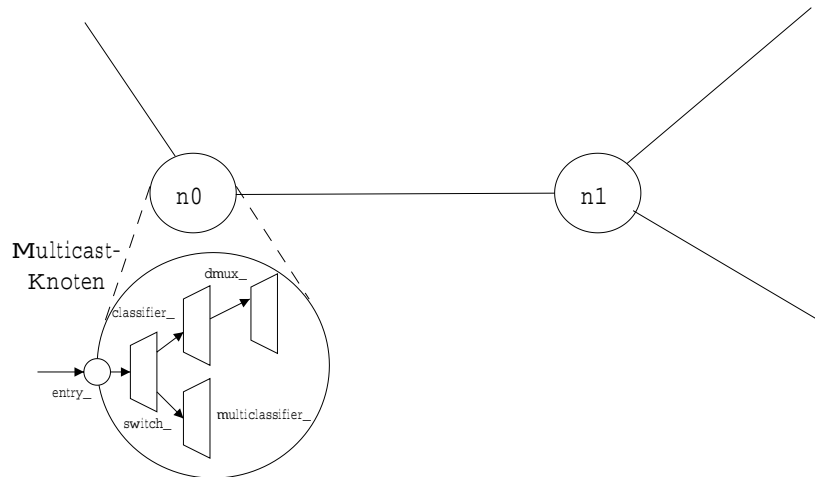


Abbildung 7.3.5: Interne Struktur eines Knotens im Simulator

Da alle *Classifier*-Objekte von einer Oberklasse erben, wurde in dieser der in 7.3.6 skizzierte Mechanismus implementiert: Beim Empfang eines Pakets wird zunächst von dem im Simulator vorhandenen Zufallsgenerator eine Zufallszahl zwischen 0 und 1 angefordert⁸. Ist diese Zufallszahl kleiner als die vorgegebene Verlustrate, wird das Paket verworfen. Nicht verworfene Pakete werden unverändert der Routingfunktion des *Classifier*-Objekts übergeben.

Durch diesen Ansatz ist dann die Wahrscheinlichkeit, dass ein bestimmtes Paket verworfen wird, proportional zur Anzahl der von ihm durchlaufenen Knoten.

Algorithmus: Simulation von Verlust im *Classifier*

```
empfang(Paket);
r := Zufallszahl();           // 0.0 <= r < 1.0
if (r < verlustrate)        // 0.0 <= r <= 1.0
then
    verwerfe(Paket);
else
    route(Paket);
end
```

Abbildung 7.3.6: Simulation von Verlust im *Classifier*

⁸Der Zufallsgenerator wird hierbei immer mit dem gleichen Wert initialisiert, so dass bei jeder Simulation die selbe Reihe von Zufallszahlen erzeugt wird. Die Simulationsergebnisse sind damit reproduzierbar.

Kapitel 8

Simulationsergebnisse

Dieses Kapitel enthält die Ergebnisse der durchgeführten Simulationen zum Aufbau von Kontrollbäumen und zur Bestimmung des Durchsatzes.

Inhaltsangabe

8.1	Aufbau des Kontrollbaums	80
8.2	Durchsatz	82

8.1 Aufbau des Kontrollbaums

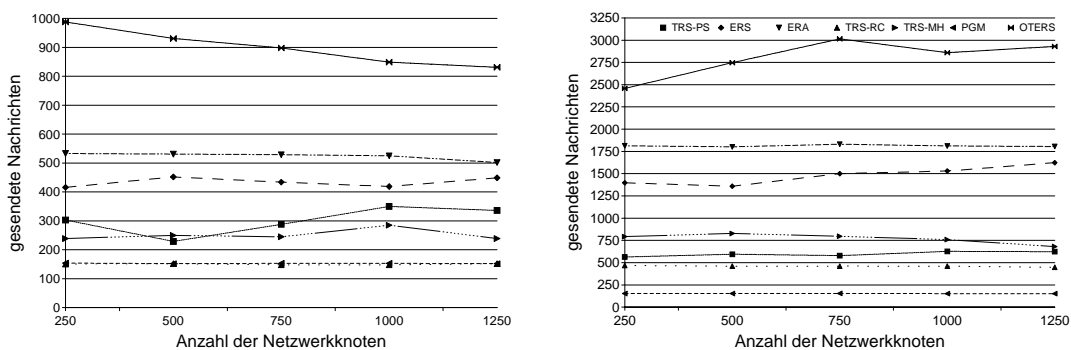
Zunächst folgen hier die Ergebnisse der Simulationen für den Aufbau des Kontrollbaums. Für grundsätzliche Überlegungen siehe auch 7.2.1. Folgende Parameter sind für alle Szenarien gleich:

Simulationszeit	200 s
Bandbreite	100 kBit/s
Paketverlust	keiner
Multicast-Routing	DVMRP

Für ERS wurde eine Wartezeit von 100 ms pro Hop gewählt. Dies bedeutet, dass der Knoten nach der Aussendung einer Suchnachricht mit $TTL = 10$ eine Sekunde auf eine Antwort wartet, bevor er eine erneute Suchnachricht mit erhöhter TTL sendet. ERA sendet alle 20 Sekunden eine Einladungsnachricht mit einer festen TTL von 127. Bei TRS-PS werden für die ERS-Suche nach einem Server (siehe 5.2.3) die gleichen Parameter verwendet wie bei ERS. In OTERS werden alle 10 Sekunden FTTP-Pakete (siehe 6.3) verschickt. Bei PGM werden die Intervalle für das Versenden der periodischen SPMs (siehe 6.6) intern berechnet und sind nicht konfigurierbar.

In allen Diagrammen ist in Richtung der x-Achse die Anzahl der Knoten im Netzwerk und in Richtung der y-Achse der Nachrichtenaufwand aufgetragen.

Die ersten beiden Diagramme in Abbildung 8.1.1 zeigen die Anzahl der gesendeten Nachrichten (wie in 7.2.1 definiert) in Abhängigkeit von der Zahl der Netzwerkknoten.



(a) 50 Gruppenmitglieder

(b) 200 Gruppenmitglieder

Abbildung 8.1.1: Anzahl der gesendeten Nachrichten in Abhängigkeit von der Netzwerkgröße.

In Abbildung 8.1.1 (a) treten 50 Empfänger der Multicastgruppe bei, in (b) sind es 200. Es ist deutlich zu sehen, dass OTERS und ERA am meisten Nachrichten versenden. Dies erklärt sich damit, dass bei OTERS ein *Designated Receiver* (siehe 6.3)

regelmässig Nachrichten mittels Subcast verschickt. Bei ERA ist es ähnlich: Wie in 5.2.2 beschrieben, versenden die Knoten des Kontrollbaums, die noch Nachfolger akzeptieren, periodische Einladungsnachrichten. Etwas weniger Nachrichten als OTERS und ERA benötigt ERS: ERS hat den Vorteil, dass keine periodischen Nachrichten verschickt werden. Allerdings braucht es in der Regel mehrere Nachrichten, bis ein Knoten des Kontrollbaums gefunden ist, der Nachfolger akzeptiert (siehe auch 5.2.1). Den geringsten Nachrichtenaufwand bezüglich der Anzahl der gesendeten Nachrichten verursachen die TRS-Varianten und PGM. Beim *Token Repository Service* ist oft nur eine einzige Anfrage an den entsprechenden Server nötig. In 8.1.1 (b) sieht man, dass bei großen Gruppen TRS-MH etwas mehr Nachrichten verursacht, da die Suche nach einem geeigneten Token etwas teurer ist. Sonst schneidet TRS-PS etwas schlechter ab, da in manchen Fällen der nächste TRS-Server mittels ERS gesucht werden muss. Das gute Abschneiden von PGM bei dieser Simulation ist damit zu erklären, dass hier nur der Sender periodische *Source Path Messages (SPM)* mittels Multicast versendet (siehe auch 6.6).

Die beiden Diagramme in Abbildung 8.1.2 zeigen die Anzahl der empfangenen Nachrichten (wie in 7.2.1 definiert) in Abhängigkeit von der Zahl der Netzwerkknoten.

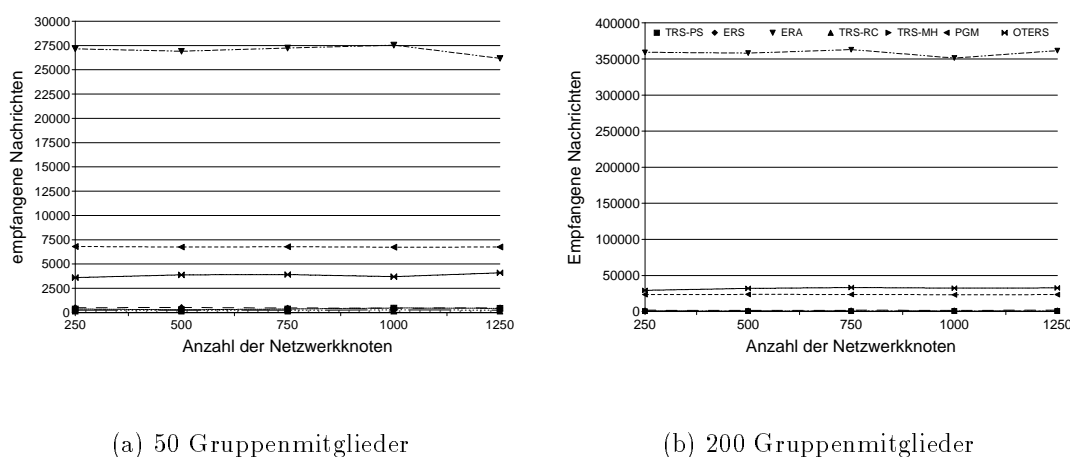


Abbildung 8.1.2: Anzahl der empfangenen Nachrichten in Abhängigkeit von der Netzwerkgröße.

In 8.1.2 (a) melden sich 50 Gruppenmitglieder, in 8.1.2 (b) 200 Gruppenmitglieder an. Wie bei den gesendeten Nachrichten zeigt sich auch hier wieder, dass Verfahren mit periodisch durch Knoten des Kontrollbaums gesendeten Nachrichten wie ERA und OTERS einen großen Aufwand verursachen. Hier schneidet aber OTERS wesentlich besser ab als ERA, weil OTERS *Subcasting* verwendet (siehe auch 6.2.1), ERA jedoch einen (limitierten) Multicast. PGM schneidet hier im Gegensatz zu oben auch recht schlecht ab, weil ja die vom Sender periodisch versendeten Nachrichten an die *ganze* Gruppe gesendet werden. Im Vergleich hierzu schneiden alle Ansätze, die keine periodischen Nachrichten aussenden, wie die TRS-Varianten und ERS, sehr gut ab.

Die beiden Diagramme in Abbildung 8.1.3 zeigen die *Packet Hops* (wie in 7.2.1 definiert) in Abhängigkeit von der Zahl der Netzwerkknoten.

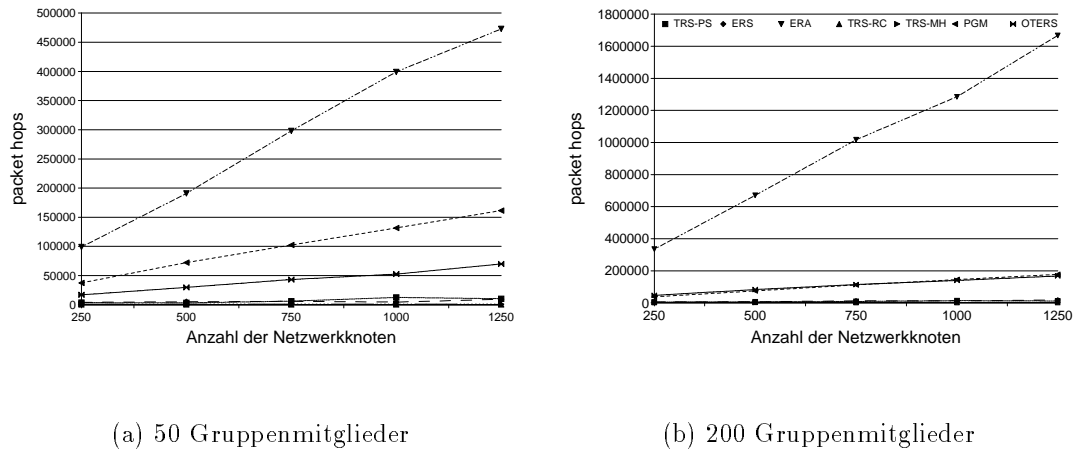


Abbildung 8.1.3: Anzahl der *Packet Hops* in Abhängigkeit von der Netzwerkgröße.

In 8.1.3 (a) melden sich 50 Gruppenmitglieder, in 8.1.3 (b) 200 Gruppenmitglieder an. Die ermittelten *Packet Hops* sind wohl das beste Maß für die verursachte Netzwerkbelastung. Man sieht, dass ERA bei weitem die größte Belastung verursacht und die *Packet Hops* mit zunehmender Netzwerkgröße auch stark ansteigen. Alle anderen Verfahren skalieren im Vergleich zu ERA wesentlich besser. In 8.1.3 (a) sieht man, dass OTERS bei kleineren Gruppen noch besser abschneidet als PGM. In 8.1.3 (b) sieht man dann, dass bei mehr Gruppenmitgliedern das Verhalten fast identisch ist. Das bessere Abschneiden von OTERS erklärt sich wieder mit der Verwendung von Subcasting im Gegensatz zum Multicasting bei PGM. Allerdings nimmt bei größeren Gruppen die Zahl der *Designated Receivers* zu, die alle periodisch ihre Nachrichten aussenden, was wohl dazu führt, dass OTERS hier seinen Vorteil wieder verliert. Auch hier zeigt sich wieder, dass alle TRS-Varianten und auch ERS eine geringe, fast konstante Last verursachen und damit ausgesprochen gut skalieren.

8.2 Durchsatz

In diesem Abschnitt folgen die Durchsatzmessungen. Für ERS, ERA und die TRS-Varianten wurden die gleichen Zeitintervalle und TTL-Werte wie bei den Simulationen zum Aufbau des Kontrollbaums gewählt. Folgende Parameter sind für alle Simulationen gleich:

Simulationszeit	200 s
Bandbreite	100 kBit/s
maximale Senderate	15 Pakete/s
Paketgröße	1024 Byte
Fenstergröße	25 Pakete
Multicast-Routing	DVMRP

Bei allen Durchsatzmessungen ist zu bemerken, dass es oft Schwankungen und Ausreißer gibt, die damit zu erklären sind, dass durch die in 7.3.3 beschriebenen Lastfiles die Gruppenmitglieder zufällig im Netzwerk verteilt werden und so für das jeweilige Verfahren einmal mehr und einmal weniger günstig liegen. Um diese Ausschläge zu vermeiden, müssten sehr viele Simulationen mit verschiedenen Netzwerktopologien und verschiedenen Lastfiles durchgeführt und Mittelwerte gebildet werden. Da die Simulationen wie schon oben erwähnt bezüglich Speicherverbrauch und Laufzeit sehr aufwendig sind, wurde hierauf im Rahmen dieser Arbeit verzichtet.

Die erste Simulation in Abbildung 8.2.1 betrachtet den Durchsatz in Abhängigkeit von der Anzahl der Netzwerkknoten. Die Simulation wurde mit 50 Gruppenmitgliedern und ohne Paketverlust durchgeführt.

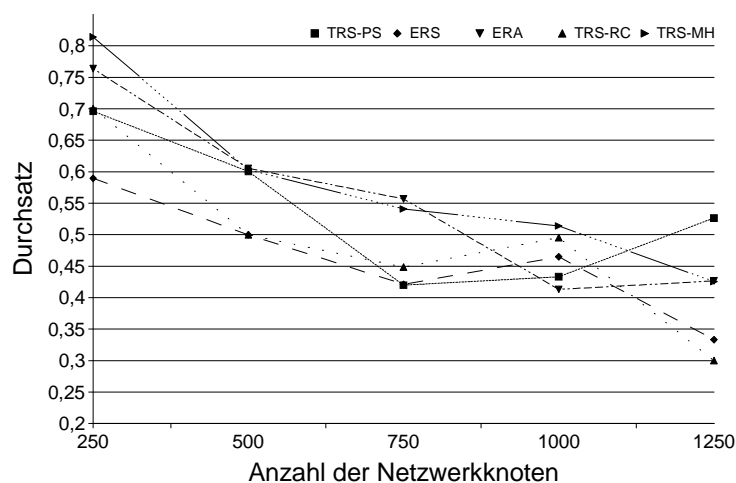


Abbildung 8.2.1: Durchsatz in Abhängigkeit vom der Anzahl der Netzwerkknoten

Abbildung 8.2.1 zeigt, dass, wie zu erwarten war, der Durchsatz mit zunehmender Anzahl der Knoten sinkt. Insgesamt erzielen die *Token Repository*-Ansätze TRS-MH und TRS-PS die besten Ergebnisse, wobei sich zeigt, dass bei geringer Knotenzahl TRS-MH und bei größeren Netzen TRS-PS besser abschneidet. Wie in 7.2.2 erläutert, sind Kontrollbäume von möglichst geringer Höhe eine Voraussetzung für gute Durchsatzraten, womit sich das gute Abschneiden von TRS-MH erklärt. In 7.2.2 wurde aber auch bemerkt, dass die Verzögerung zwischen den Knoten des Kontrollbaums ebenfalls minimiert werden sollte: Dies macht sich bei zunehmender Netzwerkgröße stärker bemerk-

bar und erklärt das dort bessere Abschneiden von TRS-PS¹. Der schlechte Durchsatz bei TRS-RC liegt daran, dass hier weder auf die Höhe des Kontrollbaums noch auf die Verzögerung optimiert wird. Das relativ gute Abschneiden von ERA überrascht etwas. Es könnte daran liegen, dass die gewählte ERA Variante mit fester TTL ähnlich wie TRS-MH zu einem recht guten Kompromiss zwischen einem Kontrollbaum geringer Höhe und geringem Abstand der einzelnen Knoten erreicht.

Abbildung 8.2.2 zeigt den Durchsatz in der Abhängigkeit der Gruppengröße. Alle Simulationen wurden mit einer Netzwerkgröße von 250 Knoten und ohne Paketverlust durchgeführt.

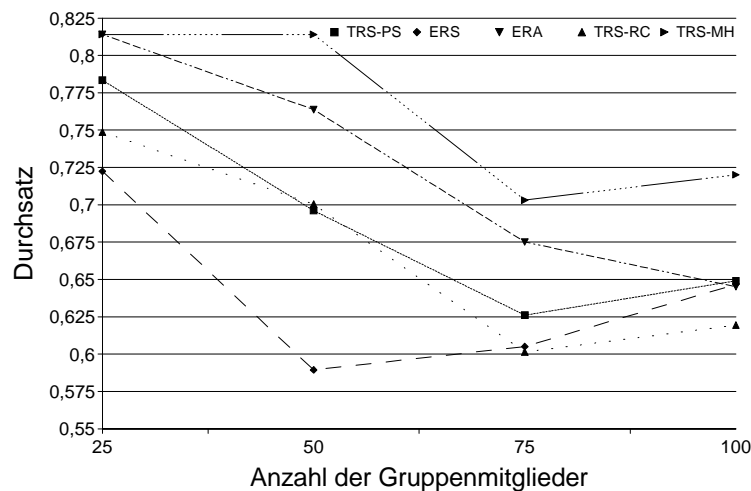


Abbildung 8.2.2: Durchsatz in Abhängigkeit von der Gruppengröße

Bei der Simulation in Abbildung 8.2.2 zeigen sich auch starke Schwankungen. Bei dieser Messung wirkt sich die zufällige Lage der Gruppenmitglieder in den Lastfiles wohl sehr stark aus. Hier wäre es sinnvoll, Simulationen mit sehr vielen Lastfiles zu machen und die Ergebnisse zu mitteln. Der Trend ist jedoch deutlich. TRS-MH kommt durch die Optimierung der Kontrollbaumhöhe mit allen Gruppengrößen sehr gut zurecht und entsprechend bricht TRS-RC stark ein.

Die Simulation in Abbildung 8.2.3 betrachtet den Durchsatz in Abhängigkeit vom Paketverlust². Die Simulation wurde mit einem Netzwerk aus 250 Knoten und 50 Gruppenmitgliedern durchgeführt.

Abbildung 8.2.3 zeigt, wie natürlich zu erwarten ist, dass bei allen Verfahren der Durchsatz bei zunehmender Verlustrate einbricht. Ohne Verlust und mit leichtem Verlust schneidet TRS-MH sehr gut ab. Bei höheren Verlustraten erzielen TRS-PS und ERS die besseren Ergebnisse. Dies erklärt sich damit, dass, wie in 7.2.2 beschrieben, der

¹Wie in 5.2.3 erklärt, versucht TRS-PS für ein neues Gruppenmitglied einen möglichst nahen Kontrollbaumknoten zu finden, der einen Nachfolger akzeptiert.

²Paketverlust wie in 7.2.2 definiert

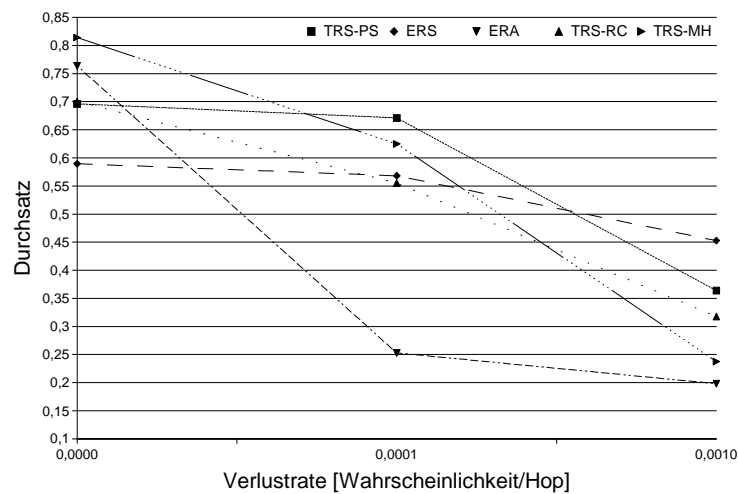


Abbildung 8.2.3: Durchsatz in Abhängigkeit vom Paketverlust

Durchsatz bei Paketverlust stark davon abhängt, wie weit ein Empfänger von seinem *Designated Receiver*, also von seinem Vaterknoten im Kontrollbaum entfernt ist. Da ERS und TRS-PS diesen Abstand minimieren, schneiden sie bei hohem Paketverlust entsprechend besser ab. Die Verschlechterung von TRS-MH bei hoher Verlustrate kommt von der Tatsache, dass die Minimierung der Höhe des Kontrollbaums hier eine Pessimierung darstellt: Die Knoten des Kontrollbaums sind dadurch in der Regel weiter voneinander entfernt, wodurch die Wahrscheinlichkeit steigt, dass ein NAK oder eine Übertragungswiederholung verloren geht.

Kapitel 9

Zusammenfassung und Ausblick

In dieser Arbeit sind eine Reihe von Ansätzen und Entwicklungen im Bereich der zuverlässigen Gruppenkommunikation analysiert und verglichen worden. Der Ansatz der Router-basierten Verfahren, Informationen aus der Netzwerkschicht für den Aufbau geeigneter Kontrollbäume und für effizientere Übertragungswiederholungen zu nutzen, klingt sehr vielversprechend. Leider haben die Simulationen zum Baufeldbau gezeigt, dass PGM und OTERS durch die Verwendung periodischer Statusnachrichten anderen Verfahren zum Aufbau des Kontrollbaums bezüglich des Nachrichtenaufwands unterlegen sind. ERS und insbesondere das *Token Repository*-Verfahren führen hier zu einer wesentlich geringeren Belastung des Netzwerks.

Bei der weiteren Untersuchung der nicht Router-basierten Verfahren anhand von Durchsatzmessungen haben sich die von den *Token Repository*-Verfahren TRS-PS und TRS-MH aufgebauten Kontrollbäume insgesamt als den anderen Verfahren überlegen gezeigt, wobei jedoch bei sehr hohen Paketverlustraten ERS die besseren Ergebnisse erzielte.

Bei diesen Ergebnissen ist zu beachten, dass alle Simulationen bewußt mit dem *Source Routed Tree* Protokoll DVMRP¹ durchgeführt wurden, welches ERS und ERA erhebliche Vorteile verschafft. Die meisten Router-basierten Ansätze setzen auch ein *Source Routed Tree* Routingprotokoll voraus.

Andere Untersuchungen [Maihöfer99] zeigen, dass die *Token Repository*-Verfahren bei einem *Shared Tree*-Verfahren wie PIM-SM im Vergleich zu ERS und ERA noch wesentlich besser abschneiden würden. Da sich zunehmend abzeichnet, dass sich PIM-SM als Multicast-Routingprotokoll im Internet durchsetzt, spricht dies sehr für den *Token Repository*-Ansatz.

Interessant wäre eine vergleichende Durchsatzmessung bei ausgewählten Router-basierten und nicht Router-basierten Protokollen. Hierzu müssten die Implementierungen so vervollständigt werden, dass auch die rein empfangerbasierten Protokolle eine Flussteuerung mit aggregierten ACKs verwenden.

¹Siehe 3.3.2.

Angesichts der Vielzahl der vorgeschlagenen Protokolle und Mechanismen für zuverlässige Gruppenkommunikation scheint es unwahrscheinlich, dass sich in naher Zukunft ein bestimmtes Verfahren durchsetzen wird. Vielmehr erscheint ein Baukastenprinzip oder *Framework* für zuverlässigen Multicast, aus dem für bestimmte Anwendungen günstige Verfahren ausgewählt werden können, wie das in [DeCleene99] vorgeschlagene *Reliable Multicast Framework (RMF)*, als vielversprechend.

Anhang A

Projektplan Diplomarbeit

A.1 Einleitung

Der Projektplan dient dem Bearbeiter und dem Betreuer als Grundlage für die Projektüberwachung und -steuerung. Er legt den Projektablauf aber nicht ein für allemal fest, sondern soll im Laufe der Arbeit ergänzt und angepaßt werden. So ist es z.B. möglich, dass anfänglich spezifizierte Arbeitspakete weiter verfeinert oder Arbeitspakete zugunsten anderer aufgegeben werden.

Der Projektplan kann in Absprache mit dem Betreuer in beiderseitigem Einverständnis in den folgenden Fällen angepaßt werden:

- bei Verzug,
- bei vorzeitigem Abschluß eines Arbeitspakets oder
- bei Gewinn neuer Erkenntnisse.

Der aktualisierte Projektplan soll in den Anhang der Ausarbeitung einfließen.

Projektbeschreibung: Bei diesem Projekt handelt es sich um eine Diplomarbeit an der Abteilung Verteilte Systeme des Instituts für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart.

Aufgabenstellung Die Aufgabe der Diplomarbeit besteht darin, Router-basierte Protokolle für zuverlässigen Multicast zu untersuchen, zu analysieren und zu simulieren. Router-basierte Protokolle haben die Besonderheit, dass sie die Funktionalität der Multicast-Router erweitern, um die Übertragungswiederholung im Fehlerfall zu optimieren. Einige Ansätze orientieren dazu den ACK-Baum an der Struktur der

Auslieferungsbaums. In der Literatur wurden verschiedene Router-basierten Protokolle vorgeschlagen. In dieser Diplomarbeit sollen mehrere Verfahren mittels Simulation miteinander verglichen werden. Der Fokus bei der Untersuchung soll dabei hauptsächlich auf dem Einfluß der ACK-Bäume liegen. Als Ergebnisse der Simulation sollen Verzögerung im ACK-Baum, Zuverlässigkeit des ACK-Baums und weitere Merkmale untersucht werden, die innerhalb der Diplomarbeit noch genauer festgelegt werden müssen.

Entwurf und Implementierung der zu erstellenden Programme sind ausführlich zu dokumentieren, die Simulationsergebnisse sollen sinnvoll aufbereitet und beschrieben werden. Der Programmcode soll leicht wartbar sein. Neben der Abgabe einer schriftlichen Ausarbeitung muss der Bearbeiter am Ende der Laufzeit einen Vortrag über die getätigte Arbeit und die erzielten Ergebnisse abhalten.

Bearbeitungszeitraum: 01.10.1999 - 31.03.2000

Betreuer: Dipl.-Inf. Christian Maihöfer

In den nachfolgenden Abschnitten werden zunächst die Arbeitspakete identifiziert und beschrieben, dann wird ein Zeitplan zu deren Durchführung festgelegt. Schließlich erfolgt die Definition der Meilensteine und der dafür zu erstellenden Dokumente.

A.2 Beschreibung der Arbeitspakete

Dieser Abschnitt umfaßt die Definition der wesentlichen Arbeitspakete, die Abhängigkeiten zwischen einzelnen Paketen und die Identifikation von kritischen Punkten, die die Durchführung des Projektes gefährden könnten.

A.2.1 Definition der Arbeitspakete

Die im Rahmen der Diplomarbeit durchzuführenden Tätigkeiten lassen sich in folgende, zusammenhängende Arbeitspakete aufgliedern:

AP1, Projektplan: Erstellung eines initialen Projektplanes

Status: abgeschlossen

AP2, Einarbeitung in Multicast Routing: genaues Verständnis für die wichtigen Verfahren, detaillierte Beschreibung

Status: abgeschlossen

AP3, Überblick über Router-basierte Ansätze: Lesen der relevanten Literatur, Beschreibung und Einordnung der Verfahren.

Status: abgeschlossen

AP4, Einarbeitung in den Netzwerksimulator NS und die Simulationsumgebung an der Abteilung VS

Status: abgeschlossen

AP5, Integration OTERS und PGM in aktuellen NS und in Simulationsumgebung

Status: abgeschlossen

AP6, Einbau einer einheitlichen Paketstatistik in NS

Status: abgeschlossen

AP7, TMTP: Übergabe eines Kontrollbaumes, Flußsteuerung und Durchsatzmessung

Status: abgeschlossen

AP8, Durchführung der Simulation

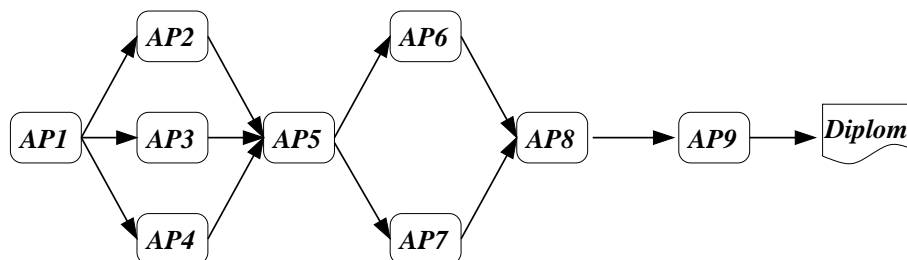
Status: abgeschlossen

AP9, schriftliche Ausarbeitung

Status: abgeschlossen

A.2.2 Abhängigkeiten der Arbeitspakete

Zwischen den Arbeitspaketen der Diplomarbeit bestehen die folgenden Abhängigkeiten:



A.3 Zeitplan

In diesem Abschnitt erfolgt die zeitliche Planung der Arbeitspakete und Meilensteine. Der Bearbeitungszeitraum ist 6 Monate oder 26 Wochen. Wenn sich der Zeitraum mehrerer Arbeitspakete überlappt, dann werden diese parallel bearbeitet. Am Ende des Bearbeitungszeitraums ist ein Puffer eingeplant worden, der für Verzögerungen oder andere unvorhergesehene Tätigkeiten genutzt werden soll.

geplanter Zeitraum	Aufwand	Arbeitspakete und Meilensteine	tatsächlicher Zeitraum
01.10.–08.10.	1 PW	AP1 Projektplan, AP2 Einarbeitung MC Routing Meilenstein 1	01.10.–08.10.
09.10.–15.10.	1 PW	AP3 Router-basierte Verf. Meilenstein 2	09.10.–15.10.
16.10.–12.11.	4 PW	AP4 Einarbeitung NS u. Simulationsumgebung Meilenstein 3	16.10.–12.11.
13.11.–24.12.	6 PW	AP5 OTERS u. PGM Meilenstein 4	13.11.–24.12.
25.12.–31.12.	1 PW	Weihnachten, Urlaub	
01.01.–14.01.	2 PW	AP6 Paketstatistik Meilenstein 5	01.01.–18.01.
15.01.–11.02.	4 PW	AP7 TMTP Meilenstein 6	19.01.–18.02.
12.02.–25.02.	2 PW	AP8 Simulationen Meilenstein 7	19.02.–07.03
26.02.–31.03.	5 PW	AP9 Schriftliche Ausarbeitung Meilenstein 8	01.03.–31.03.

(PW = Personenwoche)

A.4 Dokumente und Meilensteine

Als Ergebnis der jeweiligen Arbeitspakete sollen zu den Meilensteinen folgende Dokumente entstehen, die nach diesen einer Versionskontrolle unterliegen und nur in Absprache mit dem Betreuer geändert werden sollen.

Projektplan: Dieses Dokument.

Ausarbeitung (Teil MC-Routing): Die Kapitel der Ausarbeitung über MC (allgemein) und MC Routing

Ausarbeitung (Teil Router-bas. Verf.): Kapitel der Ausarbeitung über Router-basierte Verfahren für zuverlässigen Multicast.

Ausarbeitung (gesamt): Die gesamte Ausarbeitung für die Diplomarbeit.

Zu den Meilensteinen sind die folgenden Dokumente zu erstellen:

08.10.1999, Meilenstein 1: Projektplan, Ausarbeitung (Teil MC-Routing)

15.10.1999, Meilenstein 2: Ausarbeitung (Teil Router-bas. Verf.)

12.11.1999, Meilenstein 3: -

24.12.1999, Meilenstein 4: Lauffähige Simulation: Baumaufbau PGM u. OTERS

14.01.1999, Meilenstein 5: Einheitliche Paketstatistik f. alle Simulationen
11.02.1999, Meilenstein 6: Lauffähige TMTP-Simulation mit verschiedenen Verfahren zum Aufbau des Kontrollbaums
25.02.2000, Meilenstein 7: Simulationsdaten
31.03.2000, Meilenstein 8: Ausarbeitung

A.5 Fazit

Grundsätzlich hat sich die ursprüngliche Zeitplanung als angemessen gezeigt. Durch die langen Laufzeiten der Simulationen und die schwierige Wahl geeigneter Parameter ergab sich aber eine Verschiebung der Simulationsphase bis in die für die schriftliche Ausarbeitung vorgesehene Zeit, was sich aber dank eingeplanter Puffer noch bewältigen ließ.

Anhang B

Glossar

ACK – Acknowledgement

Nachricht, die den erfolgreichen Empfang eines Nachrichtenpakets anzeigt.

DVMRP – Distance Vector Multicast Routing Protocol

Ein in RFC 1075 beschriebenes Protokoll zur Multicast-Leitwegbestimmung.

HOP – von engl. hop: Hops, Sprung.

Metrik für den Abstand zweier Knoten im Netzwerk. Es werden die Zahl der Router auf dem Pfad zwischen beiden Knoten gemessen.

ICMP – Internet Control Message Protocol

Ein Teil des *Internet Protocol (IP)*. Das Protokoll dient dem Austausch von Kontrollnachrichten (vorwiegend Fehlermeldungen und Diagnosemechanismen) zwischen Protokollinstanzen. Ein Beispiel sind die vom bekannten *Ping*-Kommando benutzten ICMP-Nachrichten *echo request* und *echo reply*.

IGMP – Internet Group Management Protocol

Teil der Multicast-Erweiterungen des *Internet Protocol (IP)*. Das Protokoll dient dem Austausch von Informationen über Gruppenmitgliedschaften zwischen (potenziellen) Empfängern in einem lokalen Netz und ihrem (multicastfähigen) Router.

MOSPF – Multicast Open Shortest Path First

Ein in RFC 1584 beschriebenes Protokoll zur Multicast-Leitwegbestimmung. Es baut auf dem Unicast-Protokoll *Open Shortest Path First (OSPF)* auf.

NAK – Negative ACKnowledgement

Nachricht, die den Verlust eines Pakets anzeigt.

PIM – Protocol Independent Multicast

Ein Protokoll zur Multicast-Leitwegbestimmung, das nicht von einem bestimmten im Netzwerk vorhandenen Unicast-Protokoll abhängig ist.

RTT – Round Trip Time

Ein Maß für die Verzögerung zwischen zwei Knoten im Netzwerk. Die RTT zwischen

Knoten A und Knoten B ist die Zeit, die ein Paket braucht, um von A nach B und wieder zurück nach A zu gelangen.

TTL – Time To Live

Ein 8 Bit langes Feld im Header eines IP-Pakets. Der Wert wird vom Sender gesetzt und jeder Router, den das Paket passiert, reduziert den Wert um eins. Erreicht die *TTL* den Wert Null, so wird das Paket verworfen.

Tunnel

Ein Tunnel ermöglicht die Verwendung eines Protokolls in einem Netzwerk, das dieses Protokoll eigentlich nicht unterstützt. Die Protokoll-Dateneinheit (engl.: *protocol data unit*) wird dafür in eine PDU *gekapselt*, die vom zugrunde liegenden Netzwerk unterstützt wird. Man spricht in diesem Zusammenhang auch oft von *Kapselung* (engl.: *encapsulation*).

Vorwärtsfehlerkorrektur

In den Datenstrom wird Redundanz eingefügt, die es dem Empfänger erlaubt, fehlende oder fehlerhafte Pakete zu rekonstruieren (engl.: *Forward Error Correction (FEC)*).

Literaturverzeichnis

- [Ballardie93] BALLARDIE, T.; FRANCIS P.; CROWCROFT, J.: *Core Based Trees (CBT)*, Proceedings of ACE SIGCOMM '93, San Francisco, 1993
- [Braudes93] BRAUDES, R.; ZABELE, S.: *Requirements for Multicast Protocols*, RFC 1458, 1993
- [Chang84] CHANG, J.M.; MAXEMCHUK, N.F.: *Reliable broadcast protocols*, ACM Transactions on Computer Systems, vol. 2, pp.251-273, August 1984
- [Costello99] COSTELLO, A.M.; MCCANNE, S.: *Search Party: Using Random-cast for Reliable Multicast with Local Recovery*, IEEE Infocom'99, New York, 1999
- [DeCleene99] DECLEENE, B.; KUROSE, J.; TOWSLEY, D.; BHATTACHARYYA, S.; FRIEDMAN, T.; KEATON, M.; KIWIOR, D.; RUBINSTEIN, D.: *RMF: A Transport Protocol Framework for Reliable Multicast Applications*, IETF Network Working Group Internet Draft draft-decleene-rmf-01.txt, November 1999
- [Deering89] DEERING, S.: *Host extensions for IP multicasting*, Request For Comments 1112, August 1989
- [Deering95] DEERING, S.; HINDEN, R.: *Internet Protocol, Version 6 (IPv6) Specification*, Request For Comments 1883, Dezember 1995
- [Deering97] DEERING, S.; ESTRIN, D.; FARINACCI, D.; JACOBSON, V.; HELMY, A.; MEYER, D.; WEI, L.: *Protocol Independent Multicast version 2 Dense Mode Specification*, IETF Working Draft draft-ietf-idmr-pim-dm-06.txt, August 1997
- [Estrin97] ESTRIN, D.; FARINACCI, D.; HELMY, A.; THALER, D.; DEERING, S.; HANDLEY, M.; JACOBSON, V.; LIU, C; SHARMA, P.; WIE, L.: *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*, Request For Comments 2117, Juni 1997
- [Fenner97] FENNER, W.; CASNER S.: *A „traceroute“ facility for IP Multicast*, IETF Working Draft draft-ietf-idmr-traceroute-ipm-02.txt, 1997

- [Floyd97] FLOYD, S.; JACOBSON, V.; LIU, C.; MCCANNE, S.; ZHANG, L.: *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*, IEEE/ACM Transactions on Networking, December 1997, Volume 5, Number 6, pp. 784-803, Dezember 1997
- [GT-ITM] GT-ITM: *Georgia Tech Internetwork Topology Models Topology Generator*, Georgia Tech College of Computing, Georgia, <http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html>
- [Hofmann96] HOFMANN, M.: *Adding Scalability to Transport Level Multicast*, Third International COST 237 Workshop, November 25-27, 1996, Barcelona, Spain, Ed.: G. Ventre, J. Domingo-Pascual, A. Danthine: *Multimedia Telecommunications and Applications, Lecture Notes in Computer Science*, No. 1185, Page 41-55, Springer Verlag, 1996
- [Holbrook99] HOLBROOK, H.W.; CHERITON, D.R.: *IP Multicast Channels: EXPRESS Support for Large-scale Single-Source Applications*, ACM SIGCOMM '99, Cambridge, Massachusetts, 1999
- [Katz97] KATZ, D.: *IP Router Alert Option*, Network Information Center, Request For Comments 2113, Februar 1997
- [Kumar96] KUMAR, V.: *MBone: Interactive Multimedia on the Internet*, New Riders Publishing, 1996
- [Lehman98] LEHMAN, L.; GARLAND, S.; TENNENHOUSE D.: *Active Reliable Multicast*, IEEE INFOCOM'98, San Francisco, 1998
- [Levine96] LEVINE, B.N.; LAVO, D.B.; GARCIA-LUNA-ACEVES, J.J.: *The case for reliable concurrent multicasting using shared ack trees*, Proceedings of the fourth ACM International Conference on Multimedia, pages 365-376, 1996
- [Levine97] LEVINE, B.N.; GARCIA-LUNA-ACEVES, J.J.: *Improving Internet Multicast with Routing Labels*, IEEE International Conference on Network Protocols (ICNP-97), pages 241-250, 1997
- [Levine98] LEVINE, B.N.; PAUL, S.; GARCIA-LUNA-ACEVES, J.J.: *Organizing Multicast Receivers Deterministically by Packet-Loss Correlation*, Proc. Sixth ACM International Multimedia Conference (ACM Multimedia98), Bristol, UK, 1998
- [Levine98a] LEVINE, B.N.; GARCIA-LUNA-ACEVES, J.J.: *A comparison of reliable multicast protocols*, Multimedia Systems, pages 334-348, Springer-Verlag, 1998
- [Levine99] LEVINE, B.N.: *Network Support for Group Communication*, PhD Thesis, Computer Engineering, University of California, Santa Cruz, CA 95064, Juni 1999

- [Li98] LI, D.; CHERITON, D.R.: *OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol*, Proceedings of the IEEE International Conference on Network Protocols, 1998
- [Lin96] LIN, J.C.; PAUL, S.: *RMTP: A reliable multicast transport protocol*, Proceedings of the Conference on Computer Communications (IEEE Infocom), pages 1414-1424, 1996
- [Maihöfer99] MAIHÖFER, C.; ROTHERMEL, K.: *Constructing Height-Balanced Multicast Acknowledgment Trees with the Token Repository Service*, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR), Bericht 1999/15, 1999
- [Moy94a] MOY, J.: *OSPF Version 2*, Request For Comments 1583, März 1994
- [Moy94b] MOY, J.: *Multicast Extensions to OSPF*, Request For Comments 1584, März 1994
- [NS2] UCB/LBNL/VINT NETWORK SIMULATOR: *ns (version 2)*, University of California, Berkeley, <http://www-mash.cs.berkeley.edu/ns/ns.html>
- [Papadopoulos98] PAPADOPOULOS, C.; PARULKAR, G.; VARGHESE, G.: *An Error Control Scheme for Large-Scale Multicast Applications*, Submitted to IEEE Infocom '98, 1998
- [Partridge93] PARTRIDGE, C.; MENDEZ, T.; MILLIKEN, W.: *Host Anycasting Service*, RFC 1546, November 1993
- [Quinn98] QUINN, B.: *IP Multicast Applications: Challenges and Solutions*, Internet Engineering Task Force (IETF) Internet Draft draft-quinn-multicast-apps-00.txt, November 1998
- [Rothermel99] ROTHERMEL, K.; MAIHÖFER, C.: *A Robust and Efficient Mechanism for Constructing Multicast Acknowledgment Trees*, Proceedings of the Eighth International Conference on Computer Communications and Networks (IEEE ICCCN 99), 1999
- [Saltzer84] SALTZER, J.H.; REED, D.P.; CLARK, D.D.: *End-to-End Arguments in System Design*, ACM Transactions on Computer Systems, Vol.2, No.4, pages 277-288, November 1984
- [Speakman99] SPEAKMAN, T.; BHASKAR, N.; EDMONDSTONE, R.; FARIANACCI, D.; LIN, S.; TWEEDLY, A.; VICISANO, L.: *PGM Reliable Transport Protocol Specification*, Internet Engineering Task Force (IETF) Internet Draft draft-speakman-pgm-spec-03.txt, Juni 1999
- [Tanenbaum95] TANENBAUM A.S.: *Distributed Operating Systems*, Prentice-Hall, 1995

- [Tanenbaum96] TANENBAUM A.S.: *Computer Networks*, 3rd Edition, Prentice-Hall, 1996
- [Waitzmann88] WAITZMANN, D; PARTRIDGE, C; DEERING, S.: *Distance Vector Multicast Routing Protocol*, Request For Comments 1075, November 1988
- [Yavatkar95] YAVATKAR, R.; GRIFFIOEN, J.; SUDAN, M.: *A reliable dissemination protocol for interactive collaborative applications*, Proceedings of the third ACM International Conference on Multimedia, pages 333-344, 1995

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Ralf Kohler)