Universität Stuttgart

Performance-oriented Communication Concepts for Networked Control Systems

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

Vorgelegt von

Ben William Carabelli, geb. Futter

aus Mössingen

Hauptberichter:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Mitberichter:	Prof. Dr.-Ing. Frank Allgöwer
Tag der mündlichen Prüfung:	1. April 2021

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart

2022

Acknowledgements

Like any project of such scope, the completion of this dissertation would not have been possible without support. I am deeply grateful to everyone who accompanied me on this journey for their help and encouragement.

First and foremost, I would like to thank Prof. Dr. Kurt Rothermel for his mentoring and doctoral supervision. By providing me with the opportunity to work in his department, he not only made this thesis possible, but allowed me to make valuable experiences in research and teaching. Our discussions invariably provided me with helpful advice and encouraged me to challenge my assumptions.

I also want to express my gratitude to Prof. Dr. Frank Allgöwer for kindly agreeing to act as co-advisor and taking the time to review my thesis, as well as for the fruitful discussions throughout our DFG project.

Thanks also to Prof. Dr. Stefan Wagner and Prof. Dr. Marco Aiello for acting as chair and co-examiner, respectively, for the defence of this thesis.

Further thanks go to Dr. Frank Dürr, who acted as my project supervisor and would always make time in his busy schedule to discuss ideas, provide an optimistic outlook, proofread drafts, and build stuff. His dedication to research is a true inspiration. I also thank Prof. Dr. Boris Koldehofe, who provided additional support.

During my time at IPVS, I had the privilege to work alongside many friendly and talented colleagues. Besides Florian Berg and Johannes Kässinger, who deserve extra thanks as outstanding office mates, I owe a debt of gratitude to Adnan, Ahmad, Andreas, Christian, Christoph, Damian, David H., David S., Hannes, Harald, Henriette, Jonathan, Matina, Michael, Mohammad, Naresh, Otto, Patrick, Ruben, Stefan, Stephan, Sukanya, Thomas B., and Thomas K. for providing fun and companionship. Special thanks go to Annemarie Roesler, Eva Strähle, and Martin Brodbeck for their administrative support. I am also grateful to Dr. Viktor Avrutin and Dr. Michael Schanz for giving me the confidence to pursue my PhD in the first place.

Let me also take the opportunity here to acknowledge the financial support from the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) through their grants, which helped fund the research presented in this thesis.

I am grateful to my friends and family, my parents Jill and Gerhard and my sister Elena, for their love and perpetual confidence.

Finally, I would surely be lost without my fantastic wife Marina and my two lovely boys Tom and Sam. You have gone above and beyond what anyone could reasonably ask in supporting me, and give me the love and strength to tackle any obstacle. Whatever life may hold, I can count myself truly lucky.

Contents

List of Abbreviations

CAN Controller Area Network. 41, 61

CDF cumulative distribution function. 70, 79, 125

CPS cyber-physical system. 1, 8, 11, 93, 121

DOTS deterministic–opportunistic transmission slot. 6, 7, 12, 15, 22, 39, 44, 57, 130

DP dynamic programming. 6, 74

DPDK Data Plane Development Kit. 31, 32, 46

EDF earliest deadline first. 41

FIFO first-in, first-out. 14, 32, 42, 46

GCL gate control list. 15

i.i.d. independent and identically distributed. 64, 70, 80, 93, 118

IIoT industrial Internet of things. 1

ILP integer linear programming. 15

IP Internet protocol. 2, 19, 20, 23, 63, 64, 66, 78, 82, 85

LAN local area network. 3, 61, 63

LMI linear matrix inequality. 19, 26, 27, 35, 37, 129

LQ linear quadratic. 16, 20, 22, 28, 30, 34, 36, 39, 42, 47–54, 56, 62, 63, 65, 115, 125, 126, 129

LQG linear quadratic Gaussian. 89, 111, 122, 127

LQR linear quadratic regulator. 17, 22, 31, 36, 37, 43, 50, 62, 111, 122, 123

LTI linear time-invariant. 16, 22, 42, 61, 89, 110, 123

MIMO multiple-input, multiple-output. 41

MTTR mean time to repair. 118, 121

NCS networked control system. 1–9, 11–13, 16, 19–27, 29–32, 34–49, 51–53, 55–57, 61–64, 66–68, 70–73, 77–80, 83–85, 87–89, 92–94, 116, 127, 129–131

NeSTiNg Network Simulator for Time-sensitive Networking. 7

ODE ordinary differential equation. 16, 64, 77

PCP priority code point. 13, 15, 19, 44, 46

PDF probability density function. 70, 71

PTP Precision Time Protocol. 15, 69, 84

QoC Quality of Control. 3–9, 12, 16, 19, 22, 39, 42, 47, 61–63, 68, 87, 89, 110, 111, 116, 117, 121, 125, 126, 129–131

QoS Quality of Service. 2–4, 6, 7, 13, 19, 39, 57, 62, 66–69, 72, 73, 79, 81–83, 85, 129, 130

SCRaM State-Consistent Replication Management. 88, 89, 96, 99, 109, 117, 121, 125, 127, 130

SDN software-defined networking. 8, 38, 62–64, 66, 68, 84, 85

SDP semi-definite programming. 34, 39, 55, 56, 129, 130

SISO single-input, single-output. 41

SMR state machine replication. 87, 88

TDMA time-division multiple access. 3, 32, 45

TSN Time-Sensitive Networking. 3, 7, 8, 11, 14, 19, 23, 32, 39, 40, 45, 46, 61, 63

WFQ weighted fair queueing. 20, 38

List of Figures

Abstract

Networked control systems (NCS) integrate sensors, actuators, and digital controllers using a communication network in order to control physical processes. They can be found in diverse application areas, including automotive and aircraft systems, smart homes, and smart manufacturing systems in the context of Industry 4.0. Because control systems have demanding Quality of Service (QoS) requirements, the provisioning of appropriate communication services for NCS is a challenge. Moreover, the trend of steadily increasing digitization in many fields will likely lead to control applications with more complex system integration, especially in large-scale systems such as smart grids and smart cities. The proliferation of NCS in such an environment clearly depends on strong methods for integrating communication and control. However, there currently remains a gap between these two domains. On the one hand, the control-theoretic design and analysis methods for NCS have been based on simplistic and abstract network connection models. On the other hand, communication networks are optimized for conventional performance metrics such as throughput and latency, which do not readily translate into application specific Quality of Control (QoC) metrics.

The goal of this thesis is to provide performance-oriented concepts for the design of communication services for NCS. In particular, methods for scheduling and routing the traffic of NCS and increasing their reliability through replication are developed on the basis of integrated models that capture the relationship between control-relevant characteristics of communication services and the methods that are used to provide those communication services in the network. This thesis makes the following contributions.

First, we address the problem of optimally arbitrating limited communication bandwidth for a group of NCS in a shared network by designing a performance-aware dynamic priority scheduler. The resulting first scheduling policy provides asymptotic stability guarantees for each NCS and performance bounds on the joint QoC. While it is efficient to implement on the data link layer with stateless priority queueing, it requires a large optimization problem comprising all NCS to be solved initially for determining scheduler parameters. To increase the scalability, we therefore relax the scheduling problem by separating the NCS traffic into deterministic transmissions with real-time guarantees and opportunistic traffic used for QoC optimization. The resulting second scheduling policy imposes no QoS constraints on opportunistic traffic, yields less conservative stability guaran-

tees, and allows scheduler parameters to be calculated for each NCS separately and thus much more efficiently.

Second, we address the problem of optimally routing NCS traffic in networks with random latency distributions by designing a cross-layer communication service for stochastic NCS. The routing algorithm exploits trade-offs between delay and in-time arrival probabilities to find a route that provides a predefined level of QoC while minimizing network load.

Third, we address the problem of active replication for controllers in order to increase the reliability of NCS subject to crash failures and message loss. While existing replication schemes for real-time systems focus only on ensuring that no conflicting values are sent to actuators, we develop stronger consistency concepts that provide replication transparency for control systems. We present a corresponding replication management protocol that achieves high availability and low latency at low message cost, and evaluate it using physical experiments.

Zusammenfassung

Vernetzte Regelungssysteme (engl. Networked Control Systems, NCS) verbinden Sensoren, Aktoren und digitale Regler über ein Kommunikationsnetzwerk, um physikalische Prozesse zu optimieren. Sie sind in verschiedenen Anwendungsbereichen zu finden, wie z.B., im Automobil, in Luft- und Raumfahrtsystemen, Smart Homes und vernetzte Fertigungssysteme im Rahmen der Industrie 4.0. Wegen ihrer hohen Anforderungen an die Dienstgüte (engl. Quality of Service, QoS), ist die Bereitstellung geeigneter Kommunikationsdienste für NCS eine Herausforderung. Darüber hinaus wird der Trend der stetig zunehmenden Digitalisierung in vielen Bereichen voraussichtlich zu integrierten Regelungsanwendungen von steigender Systemkomplexität führen, insbesondere in weitläufigen Systemen wie Smart Grids und Smart Cities. Die Verbreitung von NCS in einem solchen Umfeld hängt entscheidend von der Verfügbarkeit leistungsfähiger Methoden zur Integration von Kommunikation und Regelung ab. Derzeit besteht jedoch noch eine Lücke zwischen diesen beiden Bereichen. Einerseits basieren gängige regelungstechnische Entwurfs- und Analysemethoden für NCS auf vereinfachten Netzwerkmodellen. Andererseits sind Kommunikationsnetze für konventionelle Leistungsmetriken wie Durchsatz und Latenz optimiert, die sich nicht ohne weiteres in anwendungsspezifische Regelgütemetriken (engl. Quality of Control, QoC) übersetzen lassen.

Das Ziel dieser Dissertation ist es, regelgüteorientierte Konzepte für den Entwurf von Kommunikationsdiensten für NCS zu entwickeln. Insbesondere werden Methoden zum Scheduling und Routing des Verkehrs von NCS und zur Erhöhung ihrer Zuverlässigkeit durch Replikation entwickelt, die auf integrierten Modellen basieren, welche den Zusammenhang zwischen regelungsrelevanten Eigenschaften von Kommunikationsdiensten und den Methoden für deren Bereitstellung im Netz abbilden. Im Einzelnen dokumentiert diese Dissertation die folgenden Beiträge.

Erstens widmen wir uns dem Problem der optimalen Zuteilung begrenzter Kommunikationsbandbreite für eine Gruppe von NCS in einem gemeinsamen Netzwerk, indem wir einen regelgüteorientierten, dynamischen Prioritäts-Scheduler entwerfen. Dieser bietet asymptotische Stabilitätsgarantien für jedes NCS und Schranken für die gemeinsame Regelgüte. Während dieser erste Scheduler auf der Sicherungsschicht (engl. Data Link Layer) durch eine Prioritätswarteschlange effizient zu implementieren ist, erfordert sie zur Bestimmung der Scheduler-

Parameter zunächst die Lösung eines großen Optimierungsproblems, das alle NCS umfasst. Um die Skalierbarkeit zu erhöhen, relaxieren wir daher das Scheduling-Problem, indem wir den NCS-Verkehr aufteilen in deterministische Übertragungen mit Echtzeit-Garantien und opportunistischen Verkehr, der zur Regelgüteoptimierung verwendet wird. Der resultierende zweite Scheduler benötigt keine QoS-Garantien für den opportunistischen Verkehr, gibt weniger konservative Stabilitätsgarantien und ermöglicht es, Scheduler-Parameter für jedes NCS separat und damit wesentlich effizienter zu berechnen.

Zweitens widmen wir uns dem Problem des optimalen Routings von NCS-Verkehr in Netzwerken mit zufälligen Latenzverteilungen, indem wir einen schichtenübergreifenden Kommunikationsdienst für stochastische NCS entwerfen. Der Routing-Algorithmus nutzt wechselseitige Abhängigkeiten zwischen Latenz und Ankunftswahrscheinlichkeit, um Routen ermitteln, die ein vordefiniertes Regelgüteniveau erreichen und gleichzeitig die Netzauslastung minimieren.

Drittens befassen wir uns mit dem Problem der aktiven Replikation von Reglern, um die Zuverlässigkeit von NCS angesichts Absturzfehlern und Nachrichtenverlusten zu erhöhen. Während bestehende Replikationsverfahren für Echtzeitsysteme lediglich sicherstellen, dass keine widersprüchlichen Werte an Aktoren gesendet werden, entwickeln wir stärkere Konsistenzkonzepte, die eine echte Replikationstransparenz für Regelsysteme bieten. Wir entwickeln ein entsprechendes Replikationsprotokoll, das eine hohe Verfügbarkeit und geringe Latenz bei niedrigen Nachrichtenkosten erreicht, und evaluieren es mit Hilfe physischer Experimente.

1 Introduction

Networked Control Systems (NCS) [HNX07; BHJ10] comprise networked sensors, actuators, and digital controllers for controlling physical processes. As such, they are of central importance for the broader field of Cyber-physical Systems (CPS) [KK12]. Depending on the application scope, NCS can be considered a special case of CPS or an enabling technology for more complex, large scale systems. They can be found in diverse application areas, ranging from automotive and aircraft systems [WYB02] to the industrial Internet of things (IIoT) or Industry 4.0 [LWA16], smart grids [SSP15], smart cities, or smart homes, to name but a few.

Figure 1.1 shows a possible basic architecture of an NCS. The *plant* is the physical process to be controlled, e.g., a car. The plant is equipped with *sensors* that measure certain features of the plant, and *actuators* that impart some effect on the plant. The *controller* is connected to the plant in a feedback loop, receiving measurements from the sensors and sending commands to the actuators. It is designed, through mathematical methods using a model of the plant, to achieve a stable closed-loop behaviour of the overall system, whereas the (open-loop) dynamical behaviour of the uncontrolled plant is usually unstable. In the example of a car, the sensors could be gyroscopes and accelerometers for measuring the yaw rate and lateral acceleration, and the actuators could be anti-lock brakes for slowing down individual wheels. The controller would then detect any deviation of the car from its desired trajectory, e.g., spinning due to loss of traction, and apply a certain amount of braking to the proper wheels to stabilize the car's motion. What makes the system in Figure 1.1 a *networked* control system is the

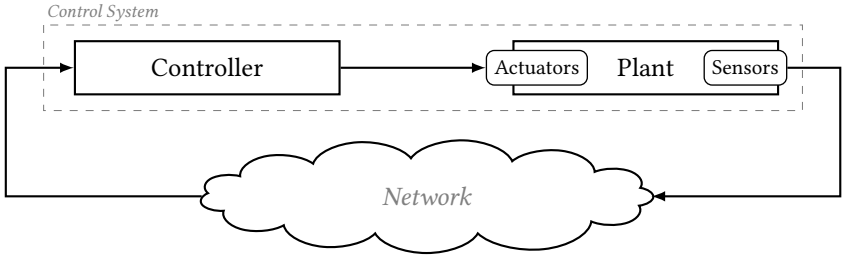


Figure 1.1: Block diagram of a simple NCS

presence of a packet-switched network in the feedback loop, in this example between sensor and controller. Therefore, the measurement signals from the sensors are necessarily transmitted to the controller in a discrete-time stream of packets. In general, of course, control signals from the controller to the actuators may also be transmitted through the network, and the controller may actually be composed of several distributed components, so NCS can come in many different connection topologies.

In contemporary NCS, a general-purpose network is often used simply as a cost-effective, flexible, and/or maintainable alternative for specialized field-bus networks or extensive dedicated wiring to close the feedback loop. However, the trend of steadily increasing digitization drives a rising demand for processes with a higher level of system integration, especially in large-scale application areas such as the aforementioned smart grids and smart cities, in which many NCS—possibly in hierarchies of systems—with vast numbers of components are distributed over a large geographical area and/or complex network topology. This need for integration concerns both the *vertical* dimension, i.e., interfaces between individual control systems and the communication service, in order to match application-specific metrics to related network metrics, and the *horizontal* dimension, i.e., coordination across control systems, in order to achieve more efficient use of the communication resources shared by several applications. While this higher level of integration offers the opportunity for widespread deployment of novel applications, it also brings challenges, in particular due to the high Quality of Service (QoS) requirements of control systems to the communication service, most prominently with regard to limited and predictable delays and packet loss rates. Ensuring this required level of QoS for multiple control systems in a flexible and scalable manner is by no means trivial, especially in Internet protocol (IP) networks.

From the control perspective, NCS have been studied intensively from various angles over the last decades. In most of those studies, the communication system is assumed to be given and modelled as random packet loss or delay. Such a given stochastic network model allows, e.g., for designing optimal state estimation with intermittent observations and determining bounds for the maximal allowed packet loss probability [Sin+04], and optimal control over networks with random packet loss [IYB06; Sch+07] and delay [Sch08] with corresponding probability bounds. While early works mostly considered simple Bernoulli models for the packet loss, progressively more sophisticated models have been employed over the years, allowing, e.g., for modelling loss of acknowledgement packets [GSC08] or Markov chain models for delay and loss [MGS13]. Unfortunately, however, stochastic modelling of loss and delay only allows for stochastic stability guarantees to be made. In a pragmatic context, such probabilistic reasoning can be sufficient, e.g., in wireless settings where a non-negligible loss probability is unavoidable.

However, strict stability guarantees in a deterministic sense are of course more useful, especially for the design of safety-critical systems. This issue is exacerbated by the fact that communication networks are designed mainly for throughput and latency, rather than providing connections with particular (and constant) stochastic loss and delay characteristics.

We are faced with several problems when designing NCS on top of existing general-purpose packet-switched networks. First, the best-effort service as provided by the current Internet, also in metropolitan and local area networks (LANs), introduces unpredictable delay and packet loss. However, as already discussed, control systems require at least a well-defined stochastic model of these network properties. Moreover, in order to enable a strict (non-probabilistic) stability and performance analysis, it is indispensable for the communication system to provide a certain degree of real-time guarantees, e.g., by ensuring maximum allowable delays and transmission intervals [Hee+10]. Second, as control systems may have unforeseen or varying communication requirements due to external disturbances, special care must be taken to utilize the limited available communication resources efficiently. In particular, optimal resource sharing should be based on control-specific performance metrics, or Quality of Control (QoC), rather than network-based metrics such as throughput. Third, the correspondence between the QoS of the network and the QoC of the closed-loop NCS is not trivial. Communication services should therefore have control-specific performance models incorporated into their design objective and offer corresponding abstractions to the control application. In summary, the integration of packet-switched networks and control systems to implement NCS requires cross-cutting research at the intersection of communications and control theory [SA11; KK12].

With respect to the problem of real-time guarantees, NCS can benefit from recent technology developments in networking. In particular, Time-Sensitive Networking (TSN), which has been recently standardized by the Institute of Electrical and Electronics Engineers (IEEE), enables real-time communication with deterministic delay bounds over standard IEEE 802.3 Ethernet through a time-division multiple access (TDMA) scheme. Besides LANs, real-time communication for larger, routed networks is also being addressed as the focus of the Deterministic Networking (DetNet) Working Group of the Internet Engineering Task Force (IETF) [Fin+18]. In this thesis, we propose to use such real-time communication technologies as a foundation for implementing control-specific communication services that allow to utilize network resources efficiently while guaranteeing stability and optimizing QoC.

1.1 Research Statement

On that basis, the problem of designing suitable communication concepts for control systems will be approached by focusing on the following problem areas.

Network and Communication Models Network models for NCS characterize the behaviour of communication services used for the transmission of measurement and control signals with respect to packet loss and latency, and play a crucial role in the system-theoretical analysis of the overall system. Many existing control design methods are based on fairly simplistic and abstract network connection models [Zha+19]. This is partly due to the fact that network models must be amenable to the formal methods of control engineering, without rendering the closed-loop model impracticably complex.

However, the systematic co-design of control and communication in NCS demands that network models be more strongly oriented towards real communication services that can be realized with modern network technologies. This implies that, rather than assuming network connections as given, integrated models shall capture the relationship between control-relevant characteristics of those connections and the decision variables and methods that are employed in the network to provide those connections. As described in the introduction, the ultimate goal is to model the relationship between the QoC and the parameters of the communication service.

Scheduling Since real control systems are usually subject to external disturbances, the requirements of the NCS regarding the QoS provided by the communication system can change dynamically. In fact, event-based controllers that sample (and transmit) based on the current state have been shown to be more bandwidth efficient than periodically sampled controllers [ÅB02]. Therefore, resource access should be carefully arbitrated with suitable scheduling policies, especially in the case where several NCS share a capacity-limited communication network. While static scheduling minimizes the mutual influence of different NCS on each other, dynamic scheduling promises optimization potential with regard to QoC, especially when taking into account the current states of the controllers and plants involved. Within this design space, performance-aware scheduling policies are to be developed.

Routing Particularly in the case of wide-area NCS, packet loss and latency of a connection can depend sensitively on the path through the network over which that connection is provided. Non-trivial trade-offs between QoC, throughput, and delay [Hee+10] imply that shortest paths are not necessarily optimal in terms of

application performance. Therefore, effective routing algorithms for NCS must integrate control-specific performance models and network-specific cost models in order to determine routes for optimal QoC.

Replication During the operation of an NCS, temporary or persistent node failures and network partitioning may occur. Active replication of controllers in the network is an obvious approach to increase the availability of the control system and thus the robustness against such unpredictable errors. However, existing replication methods are either unsuitable for real-time execution, e.g., [OL88], or provide limited consistency guarantees, e.g., [Saa+17]. Therefore, suitable consistency concepts are to be investigated in order to define necessary criteria for replication that is transparent from a control engineering point of view. On this basis, efficient replication protocols that are optimal with respect to availability and QoC are to be designed.

1.2 Contribution

In this thesis, we develop control-oriented scheduling, routing, and replication mechanisms for efficiently providing reliable communication services for control systems. These mechanisms are developed on the basis of integrated models of control and communication, which allows design goals to be specified in terms of QoC. In detail, the main contributions of this thesis are:

- We address the dynamic scheduling problem for a group of NCS sharing a dedicated network slice by designing a state-based scheduler under the assumption of a fixed network capacity. Thereby, the communication service offered to the group of applications (as a whole) provides a constant number of transmission slots each sampling period, the access to which is arbitrated through dynamic priority scheduling. The resulting scheduling policy provides asymptotic stability guarantees for each NCS and performance bounds on the joint QoC. The scheduler uses dynamic *state-based packet priorities* calculated at the sensors, which are then used for stateless priority queuing in the network, making it both scalable and efficient to implement at the data-link layer. The scheduler is designed under the assumption that each NCS comes with a given controller, thus maintaining a degree of separation of concerns between control and communication design.

  This work has been published in [Car+17]. The author of this thesis contributed approximately 94% of the content to that paper. The student thesis

[Zin16], which was co-supervised by the author of this thesis, contributed to the evaluation results.

- We relax the state-based priority scheduling problem using the communication model in [Lin+19], by assigning additional periodic guaranteed transmission slots to each NCS connection. Thereby, the communication service offered to each NCS provides a mixture of *deterministic* transmissions with reliable real-time QoS and *opportunistic* transmissions that are scheduled according to state-based packet priorities. This allows us to decouple the stability guarantees from the scheduling problem, which then only consists of (opportunistic) QoC optimization. The resulting scheduler requires no QoS guarantees for the opportunistic transmissions, provides worst-case bounds on the joint QoC, and supports incremental addition and removal of NCS from the group of scheduled applications. Moreover, a co-design method for a time-varying controller is provided, which optimizes the joint QoC under worst case assumptions.

  Part of this contribution concerning the deterministic–opportunistic transmission slot (DOTS) model has been published in [Lin+19]. The author of this thesis contributed approximately 33% of the content to that paper.

- We develop a cross-layer communication service for NCS with a probabilistic Bernoulli packet loss model, for achieving a predefined level of QoC with minimal network resource utilization. A control application can specify its required connection quality by providing a function that models the corresponding admissible trade-offs between *delay* and *in-time arrival probability* of transmissions for a given target QoC. At the network layer, the service is implemented as a routing algorithm that, given the latency distributions of links in the network topology, determines a route that can realize an admissible delay/arrival-probability trade-off with minimal network cost, based on its end-to-end latency distribution. The routing algorithm solves the corresponding constrained graph optimization problem using a dynamic programming (DP) approach. At the transport layer, the service determines the transmission interval using the maximum admissible delay.

- We address the problem of active replication for controllers in order to increase the reliability of NCS subject to crash failures and message loss. While existing replication schemes for real-time systems focus on ensuring that no conflicting values are sent to the actuators by different replicas, we develop stronger consistency concepts that provide *replication transparency* for control systems. This allows the design of the control law for a replicated controller to be treated the same as for a non-replicated controller. We present a corresponding replication management protocol that achieves

high availability and low latency at low message cost. Moreover, we show how application-specific performance metrics can be used to improve QoC in active replication.

The results on consistency concepts and the replication algorithm have been published in [CDR20]. The author of this thesis contributed approximately 85% of the content to that paper.

In the context of the research project [AR15] in which this work was carried out, the author of this thesis also co-authored works on optimal placement of operator graphs [Car+12] (with a contribution of approximately 42%) and made minor contributions to other publications concerning the DOTS model [Fal+19a; Lin+20] and the NeSTiNg simulator for TSN [Fal+19b], which are not included in this thesis.

All contributions of this thesis were developed under the careful guidance of Prof. Dr. Kurt Rothermel and Dr. Frank Dürr. They continuously supported the work by offering suggestions on the conceptual contributions and by helping to improve the presentation of results.

1.3 Research Project Context

The majority of research for this thesis was carried out in the context of the interdisciplinary project "Integrated Controller Design Methods and Communication Services for Networked Control Systems (NCS)" [AR15], which receives a Research Grant from the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) since late 2015 and is in its second period of funding at the time of writing. As a concerted effort of the Institute of Parallel and Distributed Systems (IPVS) and the Institute for Systems Theory and Automatic Control (IST) at the University of Stuttgart, this project was born out of the realization that there is a shortage of sufficiently sophisticated communication system models that are suitable for control-theoretic analysis and synthesis of NCS based on real networks. The project's research programme is guided by the following research objectives:

Network models and communication abstractions for providing suitable QoS and traffic specifications and interaction concepts. In particular, models at this level should enable cross-domain integration of methods and design objectives from control and communications, to facilitate cross-layer designs and analyses. This thesis addresses this goal in part by considering system models that integrate control systems dynamics and communication effects for assessing the overall QoC for the developed scheduling, routing, and replication methods.

Corresponding communication services for NCS to realize the developed communication abstractions, comprising methods for scheduling, routing, and controller placement, with implementation methods on the basis of software-defined networking (SDN) and TSN. This thesis addresses this goal through development of control-specific scheduling and routing methods.

Corresponding control design methods considering the salient properties of the developed network models, such as spatial and temporal correlations in service quality, weakly hard real-time guarantees, and traffic specification constraints.

Support for interdependent communication flows such as NCS with multiple spatially distributed sensors with respect to control design and analysis, as well as routing and scheduling methods.

Support for multiple interacting control loops through resource arbitration. This thesis addresses this goal by considering the optimization of joint QoC metrics for groups of (heterogeneous) control systems sharing one network through dynamic scheduling.

Increasing robustness of NCS with respect to node failures through controller replication. This thesis addresses this goal by developing a control-specific consistency concept and replication management protocol.

In a concurrent development, the DFG also established the Priority Programme "SPP 1914 – Cyber-physical Networking (CPN)" [HW16] in 2016, with the high-level goal of understanding the fundamental trade-offs between communication and control systems and developing design methods for the horizontal and vertical coordination of CPS components. To this end, 13 participating projects research a diverse range of topics including cooperative control of wireless multi-agent systems, cross-layer protocols for control-specific data rate adaptation, event-based control and resource allocation for wireless CPS, latency- and reliability-aware transport protocols, CPS benchmarking and network measurement, model-predictive control subject to communication constraints, and in-network processing for low-latency control. These activities indicate an ongoing keen interest in this research topic at the time of writing.

1.4 Structure of the Thesis

The remainder of this thesis is structured as follows. In Chapter 2, we consider the basic system model for the approaches presented in this thesis, along with some background information on communication and control. In Chapter 3, we present

state-based dynamic priority scheduler for a group of NCS sharing a fixed-capacity network. In Chapter 4, we extend the priority scheduler by adding deterministic transmissions and a controller co-design method, in order to separate stability guarantees from QoC optimization. In Chapter 5, we present an optimal routing algorithm for NCS with probabilistic network models. In Chapter 6, we present consistency conditions and a replication management protocol for fault-tolerant NCS. A summary of the contributions in Chapter 7 concludes this thesis.

2 System Model and Background

In this chapter, we present and motivate the basic system models used throughout this thesis. Since the research presented here necessarily touches on both communications and control systems, we must also discuss system models from both domains, all the while aiming to point out a unifying view. We begin by discussing the fundamental communication model from a distributed systems point of view in Section 2.1. We then refine the communication model with respect to basic reliability requirements of control systems in Section 2.2. In Section 2.3, we explain how we propose to achieve the degree of reliable communication required for this refined communication model in Ethernet using Time-Sensitive Networking (TSN) methods. Finally, we outline the application-layer model used in this thesis for discrete-time linear control systems and performance metrics thereof in Section 2.4.

2.1 Distributed System

By virtue of their architecture, NCS are fundamentally distributed systems, comprising multiple components that communicate using messages. The salient properties of distributed systems are that (a) there does not exist a global clock to which all components have instantaneous access, and (b) there is the possibility that individual components can suffer crash, omission, or timing failures [ST17] (we do not consider Byzantine failures in this thesis).

While there are many different failure and interaction models for distributed systems, the most basic classification is into *synchronous* and *asynchronous* systems. On the one hand, synchronous models make the very strong assumption of reliable communication with bounded delay, which is unrealistic for all but the most localized systems [BK14]. On the other hand, asynchronous models are mostly defined in a manner that is disconnected from real-time, in the sense that message delivery—and thereby progress and termination—is only specified in terms of *eventual* occurrence, cf. [CT96]. However, such "time-free" models are unsuitable for reasoning about CPS in general, which are inherently coupled to real-time phenomena in the physical domain, and NCS in particular, whose dynamical behaviour and key properties such as stability and performance depend delicately on the timing of measurements and control actions.

In this thesis, we therefore mainly adopt the *timed asynchronous distributed system model* proposed by Cristian and Fetzer [CF99], which posits that

a) each process has access to a local clock with a globally bounded drift rate,

b) processes are timed in the sense that methods/services have specified expected response times,

c) processes can suffer crash failures, and

d) messages can suffer omission failures.

This system model captures the properties of realistic networked systems with components possessing hardware clocks, which is a reasonable assumption for most NCS. A special case of this system model, where the clock drift rate is negligible and there is a known probability p_{loss} for omission failures, is the so-called *probabilistic synchronous* model [Dzu+16]. For the most part, we make the same assumptions here, while specifying the details of the particular system models in the respective chapters.

2.2 Deterministic–Opportunistic Transmission Model

Unfortunately, a communication model with unrestricted omission failures is unsuitable for providing control-theoretic stability and performance guarantees, since the feedback loop may be broken sporadically—or in the worst case indefinitely—, leaving the plant uncontrolled. Therefore, we recently proposed in [Lin+19] a generic communication model providing a limited degree of reliable communication suitable for NCS. We will call this the *deterministic–opportunistic transmission slot (DOTS) model* in this thesis. This model distinguishes between deterministic and opportunistic transmissions. Because deterministic transmissions are guaranteed with bounded delay, they offer a modelling tool for guaranteeing stability of NCS. Opportunistic transmissions then offer the opportunity for increasing QoC if possible. The DOTS model imposes a time-based classification into these two types of transmissions.

Consider a set of N different NCS sharing a network. We assume that there is an underlying grid of *time-slots* for all transmissions, which is determined by the common transmission period $T_s = t_{k+1} - t_k$, $k \in \mathbb{N}$. For each NCS i, we introduce two parameters s^i and d^i, where $0 \leq s^i < d^i$, which denote the phase shift and period of that system's *deterministic* transmissions, respectively. Thereby, all transmission times t_k of the deterministic transmissions of NCS i are defined by

$$k \equiv s^i \pmod{d^i}, \tag{2.1}$$

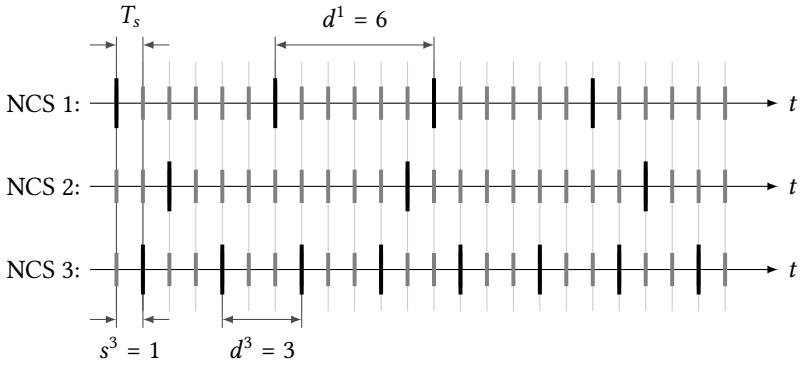


Figure 2.1: Example of deterministic/opportunistic transmission slot model for three NCS with black deterministic and grey opportunistic transmissions.

or, equivalently, $(k - s^i) \bmod d^i = 0$. We assume that deterministic datagrams are marked accordingly, for instance using the Ethernet PCP header field specified in IEEE 802.1Q, and that the corresponding transmissions are reliable with a delay of no more than T_s. (Of course, the QoS offered to deterministic transmissions can be adjusted to suit the application model.)

All other transmission slots, defined by $(k - s^i) \bmod d^i \neq 0$, are opportunistic. Individual opportunistic transmissions carry no *a priori* guarantees. However, different levels of QoS may be provided for opportunistic traffic in general. E.g., in [LA18], the opportunistic traffic of an individual NCS is considered admissible if it conforms to a token bucket specification. In this thesis, rather than considering the traffic characteristics of each control system individually, we assume that all opportunistic transmissions are handled in an aggregated fashion. More specifically, each time slot offers a certain capacity for a limited number of opportunistic transmissions. We consider scheduling mechanisms for determining how these transmissions are shared among applications.

Deterministic transmissions are handled as isolated real-time traffic with bounded queueing delay. We assume that the network provides a certain capacity q_{det} for the number of total deterministic transmissions in one time slot, and therefore assume that $\left|\{i \mid k \equiv s^i \pmod{d^i}\}\right| \leq q_{\text{det}}$. This corresponds to a resource reservation in the network subject to the constraint that the end-to-end transfer delay for a burst of q_{det} NCS datagrams does not exceed the sampling period T_s. An illustration of a possible configuration with three NCS and $q_{\text{det}} = 1$ is shown in Figure 2.1, where $(d^1, s^1) = (6, 0)$, $(d^2, s^2) = (9, 2)$, and $(d^3, s^3) = (3, 1)$.

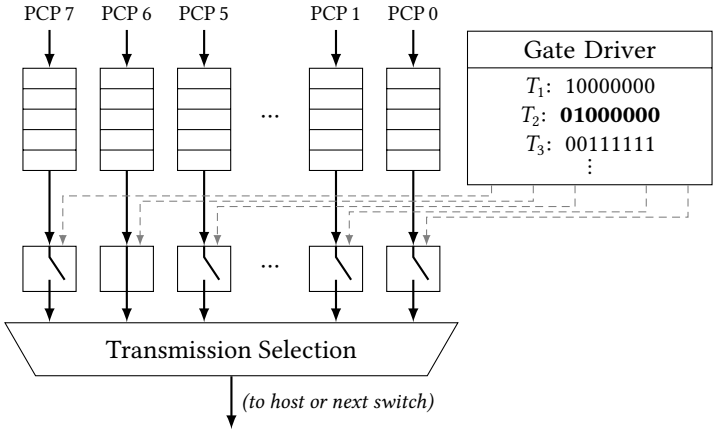


Figure 2.2: Gating and transmission selection architecture for one port of a switch complying with IEEE 802.1Qbv. Packets traverse this queueing system from top to bottom. The figure shows a situation at some time $T_2 \leq t < T_3$, where only the gate for queue 6 is open.

Deterministic transmission slots are indicated by black bars, while opportunistic slots are indicated by shortened grey bars. As can be seen in this example, there may be time slots without any deterministic transmissions. If deterministic transmissions are provisioned on a periodic basis at the common sampling period, these "leftover" timeslots could be used for opportunistic transmissions, as we propose in Chapter 4. Note that q_{det} is not required to be constant, but can be taken as time-varying without loss of generality.

2.3 Scheduling of Deterministic Transmissions

In this section, we discuss how time-triggered real-time transmission slots, in particular for deterministic transmissions, can be scheduled in IEEE 802.3 networks using standard TSN mechanisms.

While the TSN standards define multiple scheduling mechanisms, the most useful one for providing real-time transmission of time-triggered traffic is the so-called Time-aware Shaper specified in IEEE 802.1Qbv "Enhancements for Scheduled Traffic" [IEE16]. This scheduler uses so-called *gates* to control when each of the first-in, first-out (FIFO) queues associated with a particular egress port of a switch can transmit buffered packets at any given time, as shown in Figure 2.2. A switch can implement up to eight queues for egress traffic per port. During

forwarding, each frame is assigned to a queue based on its three-bit priority code point (PCP) value, which is part of the VLAN header. Buffered frames from a queue can only be transmitted over the corresponding port if the associated gate is open.

The schedule for opening and closing these gates is defined by the gate control list (GCL). Since the real-time transmission of a frame relies on all gates along its path through the network being open at the right time, the clocks of all switches must be synchronized using, e.g., the Precision Time Protocol (PTP) defined in IEEE 1588 [IEE04]. Each GCL entry comprises a time stamp and associated bit vector indicating which gates are to be opened (1) or closed (0) at that time. The schedule repeats cyclically after the last entry.

Since multiple gates might be open at the same time, the transmission selection algorithm (TSA) decides in which order to transmit the frames of open queues. For instance, the Strict Priority TSA as specified in IEEE 802.1Q serves queues by the highest PCP, such that the next frame to be transmitted would always be taken from the head of the highest-priority queue with open gate that is not empty. In the scenario depicted in Figure 2.2, only frames with PCP 6 can be transmitted in the time interval $[T_2, T_3)$, even if the higher-priority queue 7 still had outstanding frames.

The Time-aware Shaper together with the Strict Priority TSA can be used to provide real-time end-to-end service guarantees for periodic transmissions (or transmission opportunities) with a certain PCP value under the following conditions. First, the sending end-system, e.g., a sensor node, transmits at predefined time instants, e.g., as defined by the DOTS grid. Second, the transmission is forwarded along an appropriately configured path, such that only the gates of the corresponding (or lower) PCP on all egress ports along that path are open from the time when the packet arrives at the corresponding switch until it and all frames ahead of it have been transmitted to the next switch.

These conditions imply that the host transmission schedules have to be accurately synchronized to the switch schedules, and that the design of all switch schedules has to be globally coordinated, considering all real-time flows in the network in general. Different algorithms for calculating the schedules have been proposed in the literature [DN16; Cra+16; Sch+17], typically resulting in complex constraint satisfaction and optimization problems to meet delay and jitter bounds, and optimize network utilization or similar metrics. For instance, [DN16] describe an integer linear programming (ILP) formulation for calculating compact schedules for a set of flows with periodic transmissions.

2.4 Control System

The controller design for an NCS is always based on a mathematical model of the plant dynamics. In this thesis we restrict our attention to plants that are modelled as linear time-invariant (LTI) systems, whenever the study of a particular model class is necessary. A continuous-time LTI plant is modelled as an ordinary differential equation (ODE)

$$\dot{x}(t) = A_c x(t) + B_c u(t) \tag{2.2}$$

where $x \in \mathbb{R}^n$ is the state vector of the plant and $u \in \mathbb{R}^m$ is the input vector of the plant. If the state of the plant cannot be measured by sensors directly, there is also an output equation

$$y(t) = Cx(t). \tag{2.3}$$

As a measure of the QoC, we consider throughout this thesis the standard infinite-horizon linear quadratic (LQ) cost functional

$$J = \int_0^\infty x^\top(t) Q x(t) + u^\top(t) R u(t)\, \mathrm{d}t. \tag{2.4}$$

This cost metric captures the objective that the plant state x should be kept close to the origin while expending as little actuation energy as possible through u. The positive definite matrices $Q \succ 0$ and $R \succ 0$ are used to weigh individual state and input components as well as the mutual importance of these two objectives against each other.

Since networked systems necessarily exchange signals in discrete packets, our main focus is on the discrete-time variety of LTI systems. As the signals $x(t)$, $y(t)$, and $u(t)$ must be packetized in order to be transmitted over the network, the sensors are sampled periodically at time instants t_k, where we denote $T_s = t_{k+1} - t_k$ as the sampling period.

The discrete-time plant can be modelled in state-space representation as an affine difference equation

$$x_{k+1} = Ax_k + Bu_k, \tag{2.5}$$

where $x_k \in \mathbb{R}^n$ is the state at time t_k and $u_k \in \mathbb{R}^m$ is the input applied to the plant during the interval $(t_k, t_{k+1}]$. For the discretization of the system and cost matrices, in order to obtain the discrete-time model (2.5) from the continuous-time form (2.2), we refer to [LSA71].

This also leads to an equivalent discrete-time expression for the LQ cost

$$J = \sum_{k=1}^{\infty} x_k^\top Q x_k + 2x_k^\top H u_k + u_k^\top R u_k. \tag{2.6}$$

When $\left[\begin{smallmatrix} Q & H \\ H^\top & R \end{smallmatrix}\right] \succ 0$ (or, equivalently, $R \succ 0$ and $Q - HR^{-1}H^\top \succ 0$), such that J is always positive, and the pair (A, B) is stabilizable, then there exists a unique positive-definite solution P to the algebraic Riccati equation

$$P = A^\top PA - (A^\top PB + H) \cdot (R + B^\top PB)^{-1} \cdot (B^\top PA + H^\top) + Q, \tag{2.7}$$

and the optimal control input u_k^* that minimizes the cost J is given by

$$u_k^* = Kx_k, \tag{2.8}$$

$$K = -(R + B^\top PB)^{-1} \cdot (B^\top PA + H^\top). \tag{2.9}$$

This controller is called the linear quadratic regulator (LQR). Moreover, the optimal cost depending on the initial state x_0 is given by $J(x_0) = x_0^\top Px_0$ [DL71].

While this summarizes the basic control framework considered in this thesis, slightly refined (and therefore differing) control system models will be presented in the respective chapters. Those modifications concern, e.g., the incorporation of network models for loss and delay of transmissions, or the consideration of stochastic disturbances.

3 State-dependent Scheduling

3.1 Introduction

In this chapter, we assume that a dedicated "network slice" with fixed resources is available for a group of control systems. In particular, we assume that periodic real-time transmission for a certain number of datagrams is ensured, e.g., by deploying corresponding TSN schedules for a traffic class that comprises the transmissions of all participating control systems using a common PCP header value. Alternatively, the integrated services (IntServ) architecture [SPG97; Bak+01] for resource reservations in routed IP networks could be used to a similar effect. Moreover, several network virtualization technologies have been proposed which allow provisioning of such an isolated network slice with arbitrary topologies [CB10].

In such a set-up, there immediately emerges the problem of how to optimally allocate the available resources among all control systems, especially if the available bandwidth is limited compared to the totality of transmissions of all NCS considering their sampling frequency (which we assume to be uniform for simplicity). This matter is determined by the scheduling discipline which is used to arbitrate the control systems' access to the network. On the one hand, static scheduling (e.g., in a round-robin fashion) would eliminate any mutual influence of different NCS on each other. Thereby, the stability and performance of each NCS in the group can be analysed individually. On the other hand, dynamic scheduling promises optimization potential with regard to QoC, especially when taking into account the current states of the controllers and plants involved. While this introduces dynamic interdependencies between the QoS offered to each system, which prohibits individual stability and performance analyses, we begin by approaching the dynamic scheduling problem in this chapter.

We use linear matrix inequality (LMI) stability conditions for switched linear systems from [GCB08] to design a scheduler that guarantees asymptotic stability and provides performance bounds for all NCS, and show how these conditions can be generalized to accommodate concurrent transmissions. Our scheduling policy uses dynamic state-based priorities calculated at the sensors which are then used for stateless priority queuing in the network, making it both scalable and efficient to implement. Priority queuing can be found, e.g., as part of the

weighted fair queueing (WFQ) algorithm [PG93] supported by many routers.

The remainder of this chapter is organized as follows. In Section 3.2, we discuss related work. In Section 3.3, we present our system model, which comprises a set of NCS and the shared communication channel. Based on this model, we formally state our scheduling problem in Section 3.4. We then derive a scheduler under the assumption that the communication channel admits only one transmission each sampling period in Section 3.5, which we then go on to generalize for an arbitrary number of transmissions per sampling period in Section 3.6. In Section 3.7, we illustrate our approach with a proof-of-concept implementation and simulation examples. A short discussion of our results and possible avenues for future work in Section 3.8 concludes this chapter.

3.2 Related Work

An early example of a dynamic scheduler for control systems is the Maximum–Error–First policy used for the Try–Once–Discard (TOD) protocol [WYB02; NT04]. It assumes that delay-free broadcast communication is used, which may be a reasonable approximation for field-bus networks, but is an unrealistic assumption for IP networks. In [DLG10], a static periodic scheduling policy is derived for a set of different NCS, where the schedule is derived from average dwell times. In [GFB11], a stochastic RTOS scheduler for anytime control on embedded systems is studied. In [BA15], the suitability of weakly hard real-time schedulers [GQD15] has been investigated by deriving sufficient stability conditions for a single NCS, however, without directly taking network utilization and competing traffic into account.

Dynamic state-based schedulers for NCS have also been investigated. In [MH09; Ram+11], network schedulers are designed for a single delay-free control loop, investigating under which conditions a separation principle holds for the scheduler and certainty-equivalent controller design. This is extended in [RSJ13; MH14] to multiple control loops, where each sensor uses a local scheduling policy to reduce the number of transmissions, with probabilistic contention based medium access among all NCS. However, these works do not strictly follow a fixed resource constraint. In [MH14], for instance, a price for transmissions is added to the LQ cost and adapted dynamically to maintain an upper bound on the total average transmission rate. Therefore, while available bandwidth may be shared with other applications, a significant amount of bandwidth has to be over-provisioned in advance.

By contrast, we propose to use priority scheduling to utilize the reserved bandwidth as well as possible to optimize control performance. A similar approach is proposed in [Al-+13; AGL15], where a state-based dynamic priority scheduler for

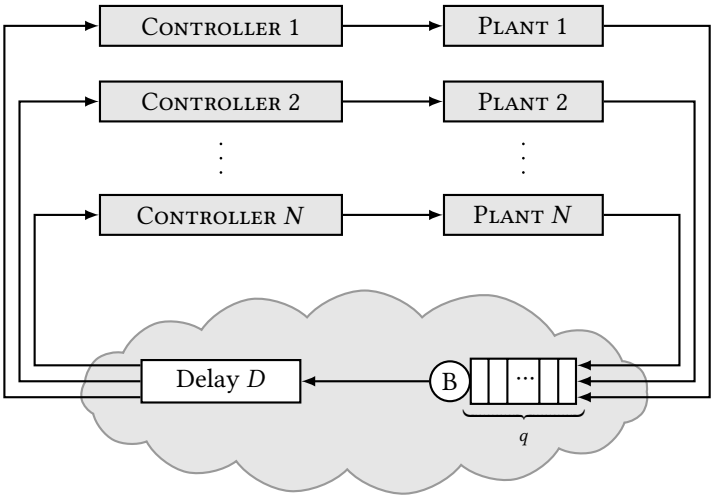


Figure 3.1: System architecture: N different NCS communicate over a shared (virtual) link with bandwidth B, delay D, and input queue of capacity q.

physically coupled NCS is derived from a quadratically structured event-triggering rule and designed together with a suboptimal controller. We, however, assume that the controllers are already given, and design a scheduler accordingly. In [AGL15], a tuning parameter is used to penalize transmissions, thereby reducing control traffic. Like the state-based scheduling strategies mentioned in the previous paragraph, this approach assumes that only one NCS can transmit at any time. We, on the other hand, provide formulations for an arbitrary fixed transmission rate (i.e., number of concurrent transmissions). Also, while [AGL15] assumes that the delays are given a priori, we use a channel abstraction conforming with [SPG97] to model the relationship between available bandwidth, overall transmission rate, and delay.

3.3 System Model

Consider a set of N different NCS, each comprising a plant and controller, where state samples are sent from sensor to controller over a (virtual) network, as shown in Figure 3.1. The network is shared by these NCS, but no other applications (i.e., no cross-traffic), and is modelled as a virtual link with bandwidth B, delay D, and an input queue of capacity q. The queue is served by a priority scheduler. This can be

considered as a special case of the DOTS model without deterministic transmission slots, where the service for opportunistic transmissions is constrained by the limited capacity in each sampling period. In the following, we describe the components of the system model—control systems, virtual link, and scheduler—in more detail.

3.3.1 Control System Model

We assume that all NCS are sampled synchronously at times t_k with a common sampling period $t_{k+1} - t_k = T_s$. Each sample of a plant's state is sent in a packet addressed to the corresponding controller, together which an attached priority value. At the same time, each controller applies a control input to the corresponding plant based on the packets that it has received so far.

In the following, the superscript index i is used to distinguish between individual control loops. The plant of NCS $i \in \{1, \dots, N\}$ is modelled as a discrete-time LTI system

$$x_{k+1}^i = A^i x_k^i + B^i u_k^i, \tag{3.1}$$

where $x_k^i \in \mathbb{R}^{n_i}$ is the state and $u_k^i \in \mathbb{R}^{m_i}$ is the input of plant i at time t_k. Upon sampling, the sensor sends a tuple $\left(x_k^i, v_k^i\right)$ over the network, where v_k^i is the packet priority used by the scheduler to dequeue packets for transmission. We will derive a function for calculating these priorities in Section 3.5. As measurements may be dropped by the scheduler due to bandwidth limitations, each controller uses the following one-step predictive control law:

$$\hat{x}_{k+1}^i = \theta(i,k) A^i x_k^i + \left(1 - \theta(i,k)\right) A^i \hat{x}_k^i + B^i u_k^i, \tag{3.2}$$

$$u_k^i = -K^i \hat{x}_k^i \tag{3.3}$$

Here, the binary arrival function $\theta(i,k)$ indicates whether the state measurement x_k^i has been successfully received at the controller by time t_{k+1} ($\theta = 1$) or not ($\theta = 0$). The controller uses the predictive state estimate $\hat{x}_k^i \in \mathbb{R}^{n_i}$ to compensate for a constant transfer delay of T_s and for dropped packets.

We assume that all plants (A^i, B^i) are controllable and that a stabilizing controller K^i is given, i.e., the matrices $A^i - B^i K^i$ are Schur. As a measure of the QoC of system i, we use the standard infinite-horizon LQ cost

$$J^i = \sum_{k=1}^{\infty} x_k^{i\top} Q^i x_k^i + 2 x_k^{i\top} H^i u_k^i + u_k^{i\top} R^i u_k^i \tag{3.4}$$

where $\begin{bmatrix} Q^i & H^i \\ H^{i\top} & R^i \end{bmatrix} \succ 0$. Therefore, a natural choice for K^i is the standard infinite-horizon LQR controller, which we use in our evaluations.

However, the controller may be arbitrary, and we assume it to be designed independently from the scheduler. Please note that we do not derive the optimal controller for this set-up. Concerning the separation of optimal control and scheduling, we refer, e.g., to [RSJ13; MH14].

3.3.2 Virtual Link Model

We assume that all NCS packets are of uniform size L, which accounts for the memory required for both the state and the attached priority value. A certain bandwidth B is allocated to the shared link, with an input queue of capacity q (in packets) to buffer packets for forwarding. Moreover, the link incurs a fixed delay D, which results from the network delay of the underlying communication service. The end-to-end delay experienced by a batch of q NCS packets is therefore $T = \frac{L}{B}q + D$. This channel model corresponds to the end-to-end service offered, for instance, by the *Guaranteed Service* class of the IntServ architecture [SPG97] for IP networks. However, as mentioned in the introduction, we assume that the fixed-capacity shared link can be implemented using TSN time-aware shaping as described in Section 2.3.

Because the transfer delay of a packet must not exceed one sampling period T_s in order to be useful to the controller, the queue must be dimensioned accordingly, such that

$$q \leq \frac{B}{L}(T_s - D). \tag{3.5}$$

Naturally, this is equivalent to a channel with a one step delay and fixed capacity q. For a given (physical) network, we may trade capacity for sampling frequency subject to (3.5).

As mentioned in the introduction, this shared link need not necessarily be physically restricted, but could also be realized through a virtualized network slice or an IP resource reservation, for instance. Therefore, the bandwidth B may also be regarded as a design parameter to adjust queue capacity and sampling period.

3.3.3 Priority Scheduler Model

Following the assumption of synchronous sampling, we also assume that all packets from one sampling period arrive at the queue simultaneously, as this yields the worst-case transfer delay. The scheduler is then responsible for servicing the q highest-priority packets, the remainder being dropped. To this end, we define the set of q-out-of-N-subsets as

$$\mathcal{S}_q = \left\{ S \subset \{1, \dots, N\} \;\middle|\; |S| = q \right\} = \bigcup_s \{ S_s \}, \tag{3.6}$$

with an arbitrary numbering $s = 1, \dots, \binom{N}{q}$. (For the simplest case $q = 1$, we choose $S_s = \{s\}$.)

Now, we can define the priority scheduling function

$$\sigma(k) = \arg\min_s \sum_{i \in S_s} v_k^i, \tag{3.7}$$

such that each NCS $i \in S_{\sigma(k)}$ receives a successful transmission in the period $[t_k, t_{k+1})$, whereas all others do not. Thereby, the scheduling function (3.7) imposes a coupling on the individual arrival indicators $\theta(i,k)$ of all NCS:

$$\theta(i,k) = \delta\big(i, \sigma(k)\big) \quad \text{with} \quad \delta(i,s) = \begin{cases} 1 & \text{if } i \in S_s \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

3.4 Problem Statement

In order to fully specify the scheduler, we need to define the priority values v_k^i for the packets generated by all NCS. The priorities must be designed such that all participating NCS remain stable under limited-capacity priority scheduling. Within these constraints, the scheduler should optimize the overall control performance. In order to formalize the problem statement, we consider the overall system using a lumped switching model that integrates all NCS models with the scheduling behaviour.

First, we rewrite the model of each individual NCS in a simpler form. Plugging (3.1)–(3.3) together, we get the following autonomous model for an individual NCS with augmented state $\mathbf{x}_k^i = [x_k^{i\top}\ \hat{x}_k^{i\top}]^\top$:

$$\begin{aligned} \mathbf{x}_{k+1}^i &= \mathbf{A}_{\theta(i,k)}^i \mathbf{x}_k^i, \\ \text{with} \quad \mathbf{A}_\theta^i &= \begin{bmatrix} A^i & -B^i K^i \\ \theta A^i & (1-\theta)A^i - B^i K^i \end{bmatrix}. \end{aligned} \tag{3.9}$$

Note that this is a switching system with two modes: $\mathbf{A}_0^i$ for open-loop and $\mathbf{A}_1^i$ for closed-loop behaviour. The cost (3.4) of the individual NCS can be rewritten as

$$\begin{aligned} J^i &= \sum_{k=1}^{\infty} \mathbf{x}_k^{i\top} \mathbf{Q}^i \mathbf{x}_k^i, \\ \text{with} \quad \mathbf{Q}^i &= \begin{bmatrix} Q^i & -H^i K^i \\ -K^{i\top} H^{i\top} & K^{i\top} R^i K^i \end{bmatrix}. \end{aligned} \tag{3.10}$$

Next, we combine all the systems into one model for the lumped state η_k of all NCS:

$$\eta_k = \left[\mathbf{x}_k^{1\top}, \mathbf{x}_k^{2\top}, \dots, \mathbf{x}_k^{N\top}\right]^\top \tag{3.11}$$

$$\eta_{k+1} = \mathcal{A}_{\sigma(k)}\eta_k, \tag{3.12}$$

$$\text{with} \quad \mathcal{A}_s = \operatorname{diag}\left(\mathbf{A}^1_{\delta(1,s)}, \mathbf{A}^2_{\delta(2,s)}, \dots, \mathbf{A}^N_{\delta(N,s)}\right)$$

This overall system switches between $\binom{N}{q}$ modes, one for every possible outcome of the scheduler $\sigma(k)$. The mode of each subsystem is determined by whether it is a member of the set $S_{\sigma(k)}$ of scheduled systems as determined by (3.8). The cost of the overall system is the sum over all NCS

$$\mathcal{J} = \sum_{i=1}^{N} J^i = \sum_{k=1}^{\infty} \eta_k^\top \mathcal{Q}\eta_k, \tag{3.13}$$

$$\text{with} \quad \mathcal{Q} = \operatorname{diag}\left(\mathbf{Q}^1, \mathbf{Q}^2, \dots, \mathbf{Q}^N\right).$$

Our goal is to choose the priorities v_k^i which determine the scheduling function $\sigma(k)$ in (3.7) such that the overall system (3.12) is asymptotically stable, while aiming to minimize the overall cost $\mathcal{J}$. Moreover, the priority v_k^i must depend solely on state information of the individual NCS i that is available at the sensor.

3.5 State-dependent Scheduler for $q = 1$

For ease of presentation, we will begin our analysis for a link capacity of $q = 1$, and later generalize the problem setting for an arbitrary queue length in Section 3.6. In this particular case, the scheduling function (3.7) simplifies to

$$\sigma(k) = \arg\min_i v_k^i.$$

We use the following sufficient condition from Geromel et al. [GCB08] to find a state-dependent, stabilizing scheduling function and performance bound for the system (3.12)–(3.13). (Other stabilizing switching designs can be found, e.g., in [LA06; LA09] and references therein.)

Theorem 3.1 ([GCB08]). *Let $\mathcal{Q} \succeq 0$ be given. If there exist a set of positive definite matrices $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_N$ and a matrix Π with entries $\pi_{ij} \geq 0$ and $\sum_{i=1}^N \pi_{ij} = 1$, $j = 1, \dots, N$ satisfying the so-called Lyapunov–Metzler inequalities*

$$\mathcal{A}_j^\top\left(\sum_{i=1}^{N} \pi_{ij}\mathcal{P}_i\right)\mathcal{A}_j - \mathcal{P}_j + \mathcal{Q} \prec 0 \tag{3.14}$$

for all $j \in \{1, \dots, N\}$*, then the switching policy*

$$\sigma(k) = \arg\min_i \eta_k^\top \mathcal{P}_i \eta_k \tag{3.15}$$

makes the origin $\eta = 0$ *of the system* (3.12) *globally asymptotically stable and*

$$\mathcal{J} = \sum_{k=1}^{\infty} \eta_k^\top \mathcal{Q}\, \eta_k < \eta_0^\top \mathcal{P}_{\sigma(0)} \eta_0, \tag{3.16}$$

i.e., the overall performance is bounded.

Note that the condition (3.14) can also be found in the context of Markov Jump Linear Systems (MJLS). More precisely, if Π were the transition probability matrix of a Markov chain, (3.14) would give mean square stability, see, e.g., [CMF05]. However, in the considered scenario, the jumps are not driven by a Markov chain but selected deterministically by the scheduler (3.15). Thus, (3.14) together with the scheduler (3.15) guarantees stability in the classical, non-stochastic sense.

Note further that finding a feasible solution to (3.14) is a non-convex problem due to the products of π_{ij} and $\mathcal{P}_i$. However, if the matrix Π is fixed, it becomes an LMI problem which can be solved efficiently using available numeric methods. Therefore, we will later propose a heuristic for determining Π using necessary stability conditions.

However, the scheduling function (3.15) depends on the full state of the lumped system, i.e. without further knowledge the states of all NCS have to be aggregated before a scheduling decision can be taken. However, our system architecture requires that packet priorities only depend on the local state.

We can rectify this by imposing some constraints on the structure of the matrices $\mathcal{P}_i$ as follows. For each NCS, we introduce two positive definite $2n_i \times 2n_i$ matrices $\mathbf{P}_0^i$ and $\mathbf{P}_1^i$ and add the restriction

$$\mathcal{P}_i = \operatorname{diag}\left(\mathbf{P}^1_{\delta(1,i)},\, \mathbf{P}^2_{\delta(2,i)},\, \dots,\, \mathbf{P}^N_{\delta(N,i)}\right) \tag{3.17}$$

to the conditions in Theorem 3.1. This allows us to rewrite the scheduling function (3.15) as follows:

$$\begin{aligned}
\sigma(k) &= \arg\min_i \eta_k^\top \mathcal{P}_i\, \eta_k = \arg\min_i \sum_n \mathbf{x}_k^{n\top} \mathbf{P}^n_{\delta(n,i)}\, \mathbf{x}_k^n \\
&= \arg\min_i \mathbf{x}_k^{i\top} \mathbf{P}_1^i\, \mathbf{x}_k^i + \sum_{n \neq i} \mathbf{x}_k^{n\top} \mathbf{P}_0^n\, \mathbf{x}_k^n \\
&= \arg\min_i \mathbf{x}_k^{i\top} \left(\mathbf{P}_1^i - \mathbf{P}_0^i\right) \mathbf{x}_k^i + \cancel{\sum_n \mathbf{x}_k^{n\top} \mathbf{P}_0^n\, \mathbf{x}_k^n}.
\end{aligned}$$

As the minimum in the expression above is independent of the last term, this is equivalent to the priority scheduler (3.7) together with the priority functions

$$v_k^i = v^i(\mathbf{x}_k^i) = \mathbf{x}_k^{i\top}\left(\mathbf{P}_1^i - \mathbf{P}_0^i\right)\mathbf{x}_k^i, \tag{3.18}$$

which depend only on the state of the corresponding NCS.

In order to use this scheduler, we must first determine whether a particular set of NCS can be stabilized under this discipline, and calculate the corresponding coefficient matrices of the priority functions. We will formulate this admission phase as an optimization problem based on the conditions in Theorem 3.1. In the following, we reformulate the Lyapunov–Metzler inequalities (3.14) and propose a heuristic for choosing the matrix Π in order to make the problem convex. The benefits of this formulation will become apparent in Section 3.6, where we show how to generalize our approach to arbitrary queue lengths.

All $\mathcal{P}_i$ as defined in (3.17) have the same block diagonal structure as $\mathcal{A}_i$ and $\mathcal{Q}$. Furthermore, as $q = 1$, the i^{th} diagonal block of $\mathcal{A}_i/\mathcal{P}_i$ is always given by $\mathbf{A}_1^i/\mathbf{P}_1^i$ (closed-loop), whereas the remaining blocks are $\mathbf{A}_0^j/\mathbf{P}_0^j$ (open-loop). This allows us to replace the inequalities (3.14) by the following set of lower-dimensional inequalities:

$$\mathbf{A}_1^{i\top}\left(\pi_{ii}\mathbf{P}_1^i + (1-\pi_{ii})\mathbf{P}_0^i\right)\mathbf{A}_1^i - \mathbf{P}_1^i + \mathbf{Q}_i \prec 0, \quad \forall_i \tag{3.19}$$

$$\mathbf{A}_0^{i\top}\left(\pi_{ij}\mathbf{P}_1^i + (1-\pi_{ij})\mathbf{P}_0^i\right)\mathbf{A}_0^i - \mathbf{P}_0^i + \mathbf{Q}_i \prec 0, \quad \forall_{i\neq j} \tag{3.20}$$

For each $i = 1, \dots, N$, we can replace all inequalities in (3.20) by a single inequality by choosing $\pi_{ij} = p_i,\ \forall_{j=1,\dots,N}$. For notational convenience we also define $m_i = \pi_{ii}$, which gives us

$$\Pi = \begin{bmatrix} m_1 & p_1 & \cdots & p_1 \\ p_2 & m_2 & \cdots & p_2 \\ \vdots & \vdots & \ddots & \vdots \\ p_N & p_N & \cdots & m_N \end{bmatrix}.$$

If we fix all m_i and p_i, then (3.19)–(3.20) become LMIs in $\mathbf{P}_{0/1}^i$. In [GCB08], the authors use an approach which is equivalent to choosing $m_i = \alpha \in [0, 1]$ and performing a line search over α to accomplish this simplification. However, since (3.14) implies that all $\sqrt{m_i}\mathcal{A}_i$, $i = 1, \dots, N$ are Schur stable [DSG15], we propose the heuristic

$$m_i = \rho(\mathcal{A}_i)^{-2} \cdot \alpha, \tag{3.21}$$

where $\rho(\cdot)$ is the spectral radius, in order to exclude infeasible values *a priori*. Because in Theorem 3.1 the matrix Π is required to be left-stochastic, i.e. with non-negative entries and columns summing up to 1, we can determine the coefficients

p_i entirely from m_i:

$$p = \left(\mathbf{1}\mathbf{1}^\top - I\right)^{-1}(\mathbf{1} - m), \tag{3.22}$$

where p and m are the corresponding column vectors of p_i and m_i. This allows us to formulate our first main result.

Theorem 3.2. *Let $q = 1$ and a set of N control systems of the form (3.1)–(3.4) be given with $\theta(i,k)$ as in (3.8). Let $\mathbf{A}_0^i$, $\mathbf{A}_1^i$, $\mathbf{Q}^i$, $i \in \{1,\dots,N\}$, be defined as in (3.9)–(3.10). If there exist a scalar $\alpha \in [0,1]$ and matrices $\mathbf{P}_0^i, \mathbf{P}_1^i \succ 0$ solving the semi-definite program*

$$\min \rho \tag{3.23}$$

$$\text{s.t. } \mathbf{P}_1^i - \rho I \prec 0, \quad \forall_i \tag{3.24}$$

$$\mathbf{P}_0^i - \rho I \prec 0, \quad \forall_i \tag{3.25}$$

$$\mathbf{A}_1^{i\top}\left(m_i\mathbf{P}_1^i + (1-m_i)\mathbf{P}_0^i\right)\mathbf{A}_1^i - \mathbf{P}_1^i + \mathbf{Q}_i \prec 0, \quad \forall_i \tag{3.26}$$

$$\mathbf{A}_0^{i\top}\left(p_i\mathbf{P}_1^i + (1-p_i)\mathbf{P}_0^i\right)\mathbf{A}_0^i - \mathbf{P}_0^i + \mathbf{Q}_i \prec 0, \quad \forall_i \tag{3.27}$$

where m and p are defined as in (3.22) and (3.21), then the scheduler (3.7) with the priority functions (3.18) makes the origin of each control system globally asymptotically stable. Moreover, the joint LQ cost is bounded by

$$\mathcal{J} \prec \rho \sum_{i=1}^{N} \|\mathbf{x}_0^i\|^2.$$

Proof. This follows from Theorem 3.1 together with the definition of $\mathcal{P}_i$ in (3.17), as shown above. To confirm the upper bound on the control cost, we can verify that

$$\mathcal{J} \prec \eta_0^\top \mathcal{P}_{\sigma(0)}\eta_0 = \textstyle\sum_{i=1}^N \mathbf{x}_0^{i\top}\mathbf{P}_{\sigma(0)}^i\mathbf{x}_0^i \prec \sum_{i=1}^N \rho\cdot\mathbf{x}_0^{i\top}\mathbf{x}_0^i,$$

due to (3.24) and (3.25). ∎

Clearly, decomposing the scheduler to enable the use of priority scheduling in the network comes at the cost of suboptimal performance compared to a centralized scheduler (3.15), as can be seen by comparing the performance bounds in Theorem 3.1 and Theorem 3.2.

3.6 State-dependent Scheduler for $0 < q < N$

In the previous section, we used the special block diagonal structure of $\mathcal{A}_i$, $\mathcal{P}_i$, and $\mathcal{Q}$ and a heuristic for Π to simplify the matrix inequalities in Theorem 3.1,

and rewrite them as an optimization problem in Theorem 3.2. There, we made the specific assumption that $q = 1$. If we drop this assumption in favour of the generalization $0 < q < N$, then rewriting inequalities (3.14) in a similar fashion yields the following two inequalities (3.28)–(3.29) instead.

$$\mathbf{A}_1^{i\top}\left(\sum_j^{i\in S_j}\pi_{jk}\mathbf{P}_1^i + \sum_j^{i\notin S_j}\pi_{jk}\mathbf{P}_0^i\right)\mathbf{A}_1^i - \mathbf{P}_1^i + \mathbf{Q}^i \prec 0, \quad \forall(i,k),\ i \in S_k \tag{3.28}$$

$$\mathbf{A}_0^{i\top}\left(\sum_j^{i\in S_j}\pi_{jk}\mathbf{P}_1^i + \sum_j^{i\notin S_j}\pi_{jk}\mathbf{P}_0^i\right)\mathbf{A}_0^i - \mathbf{P}_0^i + \mathbf{Q}^i \prec 0, \quad \forall(i,k),\ i \notin S_k \tag{3.29}$$

It is easy to verify that (3.19)–(3.20) are a special case thereof.

As before, we can reduce this to a pair of inequalities for each NCS $i = 1, \dots, N$. For this purpose, the coefficients π_{jk} of the matrix Π must satisfy the following conditions:

$$\textstyle\sum_{\{j|\ i\in S_j\}} \pi_{jk} = m_i, \quad \forall(i,k),\ i \in S_k \tag{3.30}$$

$$\textstyle\sum_{\{j|\ i\in S_j\}} \pi_{jk} = p_i, \quad \forall(i,k),\ i \notin S_k \tag{3.31}$$

We assume again that the vector m is given, e.g., using (3.21). As Π now has additional degrees of freedom and its entries are not directly determined by m and p as they were in the previous section, we need to use a different heuristic to determine a feasible p. For instance, we can use the following optimization problem:

$$\min_{\Pi,p} \quad \operatorname{tr}(\Pi) \tag{3.32}$$

$$\text{s. t.} \quad \pi_{ij} \geq 0, \quad \forall_{i,j} \tag{3.33}$$

$$\textstyle\sum_i \pi_{ij} = 1, \quad \forall_j \tag{3.34}$$

$$\pi_{ii} \leq \rho(\mathcal{A}_i)^{-2}, \quad \forall_i \tag{3.35}$$

$$(3.30), (3.31), \tag{3.36}$$

where the first two constraints are required by Theorem 3.1. The third constraint is equivalent to the necessary condition that $\sqrt{\pi_{ii}}\mathcal{A}_i$ must be Schur, which is also our reasoning behind minimizing the trace of Π.

With (3.30)–(3.31) the inequalities (3.28)–(3.29) become equivalent to the linear inequalities (3.26)–(3.27) in Theorem 3.2. This allows us to generalize the same result to an arbitrary queue length.

Theorem 3.3. *Let $0 < q < N$ and a set of N control systems of the form (3.1)–(3.4) be given with $\theta(i,k)$ as in (3.8). Let $\mathbf{A}_0^i$, $\mathbf{A}_1^i$, $\mathbf{Q}^i$, $i \in \{1, \dots, N\}$, be defined as in*

(3.9)–(3.10). *If there exist a scalar* $\alpha \in [0,1]$ *and matrices* $\mathbf{P}_0^i, \mathbf{P}_1^i \succ 0$ *solving the semi-definite program* (3.23)–(3.27)*, where* m *and* p *are defined as in* (3.21) *and* (3.32)–(3.36)*, then the scheduler* (3.7) *with the priority functions* (3.18) *makes the origin of each control system globally asymptotically stable. Moreover, the joint LQ cost is bounded by* $\mathcal{J} < \rho \sum_{i=1}^{N} \|\mathbf{x}_0^i\|^2$.

Proof. Analogous to the previous section, it is straightforward to verify that

$$\sigma(k) = \arg\min_i \eta_k^\top \mathcal{P}_i \eta_k = \arg\min_s \sum_{i \in S_s} \mathbf{x}_k^{i\top} \left(\mathbf{P}_1^i - \mathbf{P}_0^i\right) \mathbf{x}_k^i.$$

From there, the same line of argument as for Theorem 3.2 holds. ■

Interestingly, this shows that the scheduling problem for any queue length can be reduced to the same semi-definite programming problem, provided the parameters p are chosen appropriately according to the queue length.

Note that our previously defined scheduler requires both the plant state x_k^i and the controller state $\hat{x}_k^i$ to be known to calculate the priorities $v^i(\mathbf{x}_k^i)$. This can be accomplished by maintaining a copy of the controller state recurrence (3.2) at the sensor, for which $\theta(i,k)$ must be known, e.g., by sending acknowledgements from the controllers to the sensors, cf. [RSJ13]. However, acknowledgements would have to be delivered reliably, e.g., by reserving an additional virtual link. Also, sending an acknowledgement of size L_{ACK} introduces an additional delay, which must be modelled by modifying the propagation delay D, e.g., to $D' = 2D + \frac{L_{\text{ACK}}}{B}$ under the assumption of a symmetric duplex channel.

Alternatively, we can impose an appropriate structure on the optimization problem (3.23)–(3.27). If we add the following constraints

$$\mathbf{P}_1^i = \begin{bmatrix} X_1^i & Y^i \\ Y^{i\top} & Z^i \end{bmatrix}, \quad \mathbf{P}_0^i = \begin{bmatrix} X_0^i & Y^i \\ Y^{i\top} & Z^i \end{bmatrix}, \tag{3.37}$$

then the priority functions become $v^i(x_k^i) = x_k^{i\top} \left(X_1^i - X_0^i\right) x_k^i$, which depend only on the plant state. A similar approach is proposed in [GCB08] in a different set-up to design a switching output feedback controller. Of course, the corresponding optimization problem is more conservative. However, it allows us to achieve a higher bandwidth utilization than using acknowledgements.

3.7 Evaluation

To demonstrate the feasibility of our approach, we ran experiments using real networking hardware with a set of simulated NCS and a proof-of-concept implementation of the priority scheduler. We also show a set of pure simulation

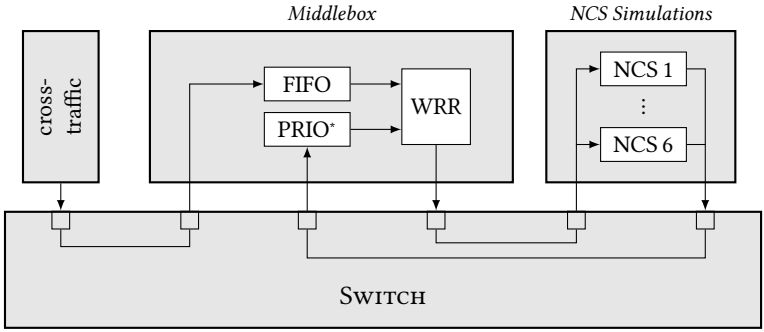


Figure 3.2: Networking testbed set-up for scheduler evaluation.
*) Priority scheduling and deficit round robin scheduling were implemented for comparison.

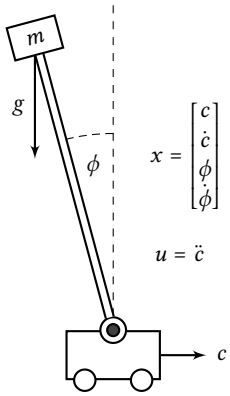
examples to illustrate the relationship between queue size, bandwidth utilization and control cost.

3.7.1 Proof-of-concept Implementation

We ran a proof-of-concept evaluation in a networking test-bed consisting of commodity machines (Intel Xeon E5-1650) with 4 × 10 Gbps Ethernet network interface cards, connected to a 10 Gbps Ethernet switch (Edge-Core AS5712-54X). The set-up of our test-bed is shown in Figure 3.2.

A set of $N = 6$ NCS were simulated on one of the machines, with outgoing measurement packets from the simulated sensors of all NCS instances being sent over one network interface, and incoming packets being received on a different interface. In order to improve throughput and reduce unpredictable delays, we used the Data Plane Development Kit (DPDK) [Lin] for sending and receiving packets from the simulation instances, bypassing the operating system's protocol stack. We used the same plant model (3.1) for all 6 NCS simulation instances, namely and inverted pendulum on a cart as shown in Figure 3.3, with a standard LQR controller. The sampling period was chosen at $T_s = 50$ ms. The pendulum for NCS 1 was started at an initial angle of $\phi = 35^\circ$, while all others were started at the origin $\phi = 0^\circ$.

A second machine was dedicated to producing cross-traffic approximately at line rate on a dedicated network interface. We use this cross-traffic to demonstrate that our proof-of-concept implementation realizes a dedicated virtual link to isolate NCS traffic from the effects of traffic from other applications, which is one of our initial assumptions.



$$x = \begin{bmatrix} c \\ \dot{c} \\ \phi \\ \dot{\phi} \end{bmatrix} \qquad u = \ddot{c}$$

$$A = \begin{bmatrix} 1 & 0.05 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1.015 & 0.05 \\ 0 & 0 & 0.603 & 1.015 \end{bmatrix} \qquad B = \begin{bmatrix} 0.001 \\ 0.05 \\ 0.002 \\ 0.062 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.05 & 0.001 & 0 & 0 \\ 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0.101 & 0.003 \\ 0 & 0 & 0.003 & 0 \end{bmatrix} \qquad R = 0.05$$

$$K = \begin{bmatrix} -0.811 & -1.641 & 26.074 & 7.517 \end{bmatrix}$$

Figure 3.3: Inverted pendulum model for scheduler evaluation. System parameters for (3.1)–(3.4) are shown on the right.

A third machine was used to host a software middlebox implementation realizing both the virtual link provisioning and priority scheduling. The middlebox was also implemented using DPDK in order to eliminate latencies introduced by the operating system's networking stack. Three of the middlebox's network interfaces were connected to the switch: one for incoming NCS traffic, one for incoming cross-traffic, and one for outgoing traffic. The middlebox maintains two input queues: a priority queue with capacity $q = 2$ for NCS traffic, and a FIFO queue for all other traffic. Packets received on the cross-traffic interface are directly placed in the FIFO queue, while NCS packets are parsed to extract the priority value from their payload, before being placed at the appropriate position in the priority queue, with the lowest priority packet being dropped as necessary. The two queues are served in a weighted round robin fashion, where the priority queue is fully served every $T_s = 50\,\text{ms}$ and the FIFO queue is served during the remaining time. This corresponds to the TDMA scheme offered by TSN. All outgoing traffic is transmitted on the same interface. The switch classifies all frames received from the middlebox into NCS traffic and cross-traffic by address header fields. NCS traffic is forwarded to the NCS simulation node, while cross-traffic is discarded at the switch. In order to compare our state-based dynamic scheduler to a fair static scheduler, we also implemented an alternative middlebox, with the difference that the queue for NCS traffic is not served using packet priorities, but using a deficit round robin (DRR) protocol. Thereby, each NCS can successfully transmit a packet once every three sampling periods, and all achieve the same bandwidth.

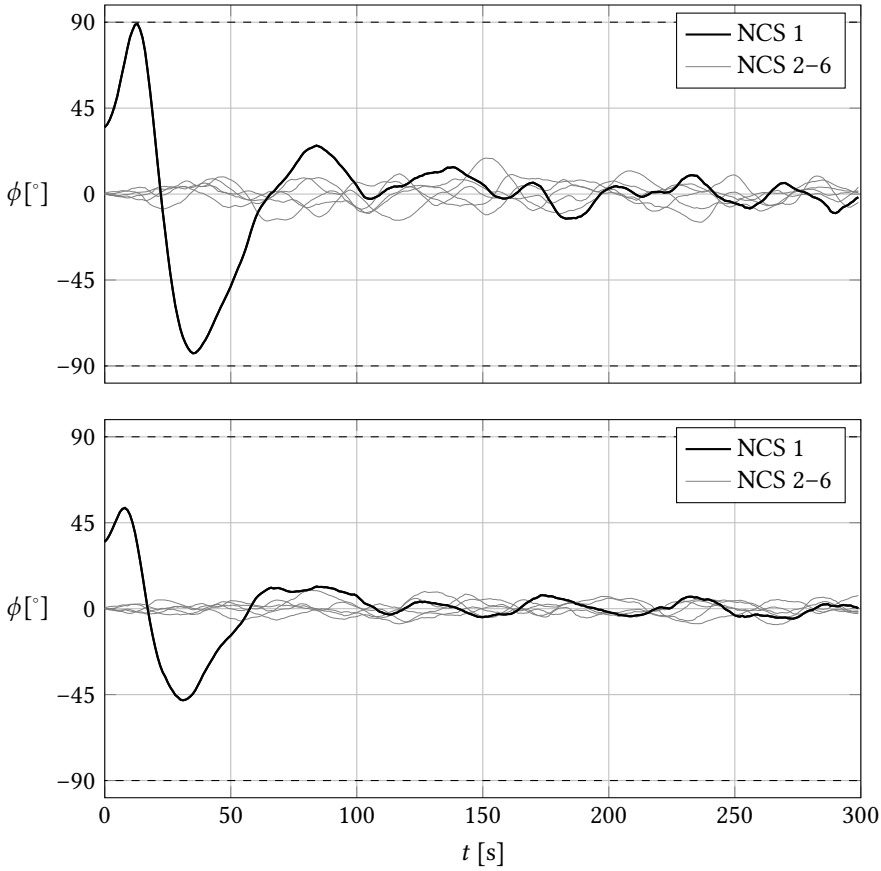


Figure 3.4: Time series of pendulum angles from evaluation of proof-of-concept implementation with $N = 6$, $q = 2$, and $T_s = 50\,\text{ms}$; top: round robin scheduling; bottom: state-based priority scheduling.

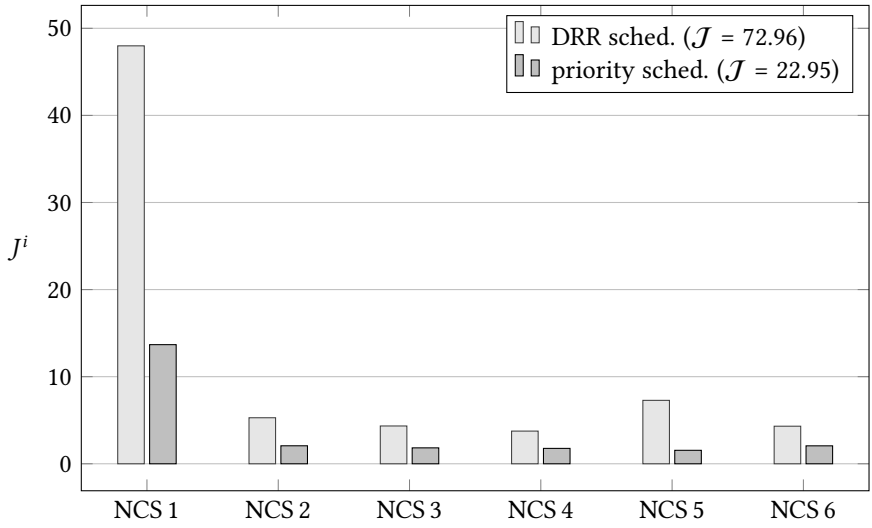


Figure 3.5: LQ cost of individual NCS for evaluation of proof-of-concept implementation with $N = 6$, $q = 2$, and $T_s = 50$ ms, comparing deficit round robin scheduling and state-based priority scheduling. Joint LQ cost $\mathcal{J}$ shown in legend.

We ran evaluations for both configurations: once using naive deficit round robin and once using our state-based priority scheduling for NCS traffic. Figure 3.4 shows time series for the angle ϕ of all six simulated pendulums. Using priority scheduling instead of round robin scheduling reduces the peak angles of NCS 1 to approximately 55%, and reduces the variance of angles significantly for all NCS.

Figure 3.5 shows the joint and individual LQ cost for comparison. Using state-based priority scheduling reduces the joint cost $\mathcal{J}$ by 68% compared to round robin. Moreover, the cost is more evenly distributed. While the stretch between the worst and best performing NCS is $\frac{\max_i J^i}{\min_i J^i} = 12.8$ with round robin, it is only 8.8 with state-based priority scheduling.

3.7.2 Runtime Evaluation

To evaluate the complexity of solving the optimization problem (3.23)–(3.27) required for calculating the priority function parameters, we used the Julia [Bez+18] optimization toolbox JuMP [DHL15] together with the commercial semi-

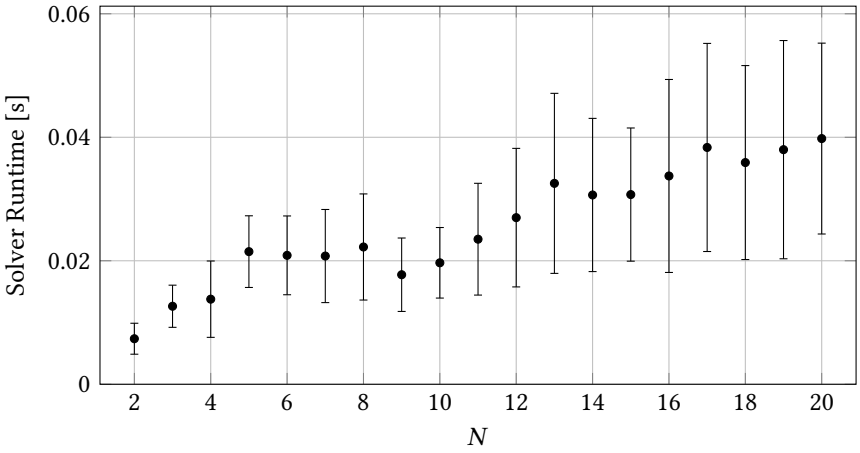


Figure 3.6: Runtime evaluation for state-dependent scheduling

Table 3.1: Problem size of state-dependent scheduler compared to [AGL15]

Approach	#LMIs	size(LMI)	#vars
ours	$4N$	$2n \times 2n$	$2N(2n^2+n)+1$
[AGL15]	$> (N+1)^2$	$4N(n+m) \times 4N(n+m)$	$> N^3(n+m)^2$

definite programming (SDP) solver Mosek [MOS18]. We solved the problem with increasing size, for $N = 2, \dots, 20$. In Figure 3.6, the average solver times are shown, with standard deviations over 50 runs indicated by error bars. The results suggest that the time required to solve the optimization problem scales benignly with an increasing number of systems.

Also, in Table 3.1 we compare the size of the optimization problem in Theorem 3.2 with that proposed in [AGL15] for a set of N systems, all with state dimensions n and input dimensions m, in terms of the number of LMI constraints, dimensions of LMI constraints, and number of scalar decision variables. While [AGL15] jointly designs a suboptimal controller in the process, our approach clearly remains more practical from a computational point of view as the number of participating NCS grows.

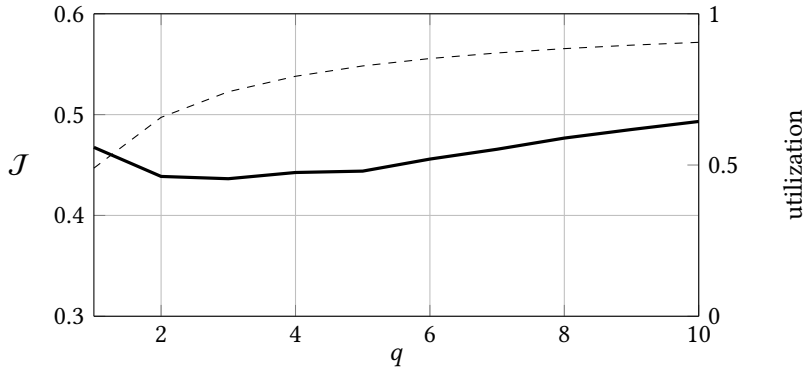


Figure 3.7: Simulations for varying queue capacity q: joint cost $\mathcal{J}$ (thick) and bandwidth utilization (dashed).

3.7.3 Simulation Example

To illustrate the effects of queue dimensioning, we show some simulation results for a fixed set of NCS and network model with varying q. We simulate a link with bandwidth $B = 10\,000\ \frac{\text{bit}}{\text{s}}$ and delay $D = 20\,\text{ms}$, that is shared by $N = 10$ NCS with identical (continuous) plant dynamics

$$\frac{\mathrm{d}}{\mathrm{d}\,t} x^i(t) = \begin{bmatrix} 0 & 1 \\ -2 & 2 \end{bmatrix} x^i(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u^i(t) + w^i(t),$$

initial conditions $x^i(t_0) = \begin{bmatrix} 1, 1 \end{bmatrix}^\top$, LQR controller for $Q = I$ and $R = 0.1$, and additive white noise $w^i(t) \sim \mathcal{N}\left(0, 10^{-3} I\right)$. We assume the packet size to be $L = 192\,\text{bit}$, which corresponds to two IEEE double-precision floating point values for the state and one for the priority. The overall system with the state-based scheduler is simulated for $q = 1, \ldots, N$ over a time period of 100 seconds. In each simulation, the systems are discretized to the minimum possible sampling time according to the link model, i.e. $T_s = \frac{L}{B} q + D$.

The results, averaged over 50 realizations of the simulation, can be seen in Figure 3.7. The plot shows the joint LQ cost $\mathcal{J}$, together with the bandwidth utilization, which is given by the ratio of the transmission rate $\frac{qL}{T_s}$ to the available bandwidth B. Note that the case $q = 10$ corresponds to a static equal-bandwidth schedule. We can see that the cost decreases with decreasing q, corresponding to more dynamic scheduling, but increases again towards $q = 1$. This can be attributed to a lower bandwidth utilization, as the propagation delay D dominates the sampling period for small q.

For $q = 1$, the following cost coefficient matrix is found:

$$\mathbf{P}_1^i - \mathbf{P}_0^i = \begin{bmatrix} -8.9 & 6.6 & 8.9 & -6.6 \\ 6.6 & -8.1 & -6.6 & 8.1 \\ 8.9 & -6.6 & -8.9 & 6.6 \\ -6.6 & 8.1 & 6.6 & -8.1 \end{bmatrix}$$

3.8 Summary and Discussion

In this chapter, we addressed the optimal scheduling problem for a set of NCS sharing a dedicated network slice. We introduced a switched model of the overall system with a limited-capacity queue model for the communication channel. Based on LMI stability conditions for switched linear systems from [GCB08], we first designed a state-based priority scheduler for a channel capacity of one transmission per sampling period. We then generalized our scheduler design to allow an arbitrary number of NCS to transmit concurrently within one sampling period. The resulting scheduling policy guarantees performance and asymptotic stability of all NCS, and only requires stateless priority queuing in the network, making it both scalable and efficient to implement.

To conclude this work, we would like to discuss some aspects of our results. We designed a scheduler under the assumption that all NCS come with a given controller. Of course, even if we assume that the standard LQR controller is used, which is optimal for periodic sampling, it is not necessarily the optimal controller when subjected to our scheduling policy. The co-design of an optimal controller, e.g. as attempted in [AGL15], for our scheduler is an interesting topic for future work. Furthermore, it could be studied how to choose the queue length for a given available bandwidth and network delay, such as to optimize the overall control performance. Also, while we considered priority scheduling for full state feedback here, the output feedback case can be studied by adding restrictions to the LMI constraints, as proposed in [GCB08].

Our approach does not provide isolation between the traffic of individual NCS. While this allows us to utilize the available bandwidth to improve overall performance, it opens up the opportunity for one or more NCS to use more of their "fair" share to the detriment of all others, possibly to the point of instability. Apart from malicious priority inflation, modelling errors could also lead to this kind of behaviour. One possible solution is to use traffic shaping to limit the bandwidth available to each NCS. The consideration of additional shaping constraints is a topic for future work.

In our system model, we make some assumptions that should also be discussed briefly. First, sampling times and therefore packet arrival times of all NCS are synchronized. If this assumption is violated, i.e., sampling times of different

NCS are phase shifted or packets experience different delays between sensor and scheduler, then any low-priority packet arriving early could impose an additional queuing delay on the remaining NCS traffic and might ultimately cause a higher-priority packet to be dropped. Moreover, we did not account for priorities from different (overlapping) sampling periods to be compared at the scheduler.

Second, the communication channels of all NCS are modelled by one shared link. However, in a realistic application scenario, it should be assumed that the traffic of different NCS is routed over overlapping multi-hop paths. This means that a scheduling decision may be required at every hop. In principle, the same scheduler could be implemented at all network elements, since the priorities are preserved under scheduling of different packet subsets in an arbitrary order. However, even if the sampling times are synchronized, this is not necessarily true for the arrival times at schedulers in the network, which has been discussed above. Also, it may lead to suboptimal resource utilization, since unnecessary constraints may be imposed on traffic flows with mutually disjoint network paths. How our approach can be extended to address these issues is to be investigated in future work.

Finally, there are some practicalities to consider concerning the implementation of our proposed scheduler in the network. In Section 3.5 we made the case that our approach requires only a priority queue at the data link layer. While this is also a part of available scheduling disciplines such as WFQ [PG93], we would have to be able to subvert the mechanism by which priorities are assigned to packets. Furthermore, the priorities would have to be extracted from the packets' payload using deep packet inspection, or be mapped to an available packet header field, which would introduce rounding errors. However, the case for opening up programming interfaces to customize scheduling has already been made by others, e.g. [Siv+13]. Following a related development in the recent success of SDN and the OpenFlow protocol [McK+08], which enables applications to define flexible routing and forwarding policies, there is hope that such interfaces may become available in the future.

4 Opportunistic Scheduling

4.1 Introduction

In Chapter 3, we presented a priority scheduling policy for NCS traffic, where packet priorities for measurement packets are dynamically calculated at the sensors as a function of the current state of each control system. By formulating the scheduling problem using a switched system model, we were able to determine parameters for the individual priority functions by solving a SDP that provides sufficient conditions for the stability of all NCS and optimal bounds for the overall control performance in terms of joint LQ cost.

However, this approach has some drawbacks. While it allows for sharing the available bandwidth with a high degree of flexibility between a set of NCS, the fact that transmissions are scheduled purely based on packet priorities implies that no explicit bandwidth guarantees are given to any individual NCS. Therefore, the stability conditions for a set of NCS depend implicitly on the priority scheduling policy, and tend to be very conservative. Specifically, the corresponding SDP optimization problem quickly becomes infeasible for larger numbers of involved control systems, depending on their dynamics. Moreover, as also in [AGL15], it is not possible to incrementally add/remove NCS to/from a network without solving the global optimization problem again to find a new scheduling configuration. As a result, pure priority scheduling does not scale well.

In this chapter, we therefore propose to leverage the DOTS model described in Section 2.2 in order to decouple the stability guarantees from the QoC optimization. While the deterministic traffic of an NCS is provided with real-time guarantees and isolated using, e.g., TSN technologies, the opportunistic traffic of all NCS can be multiplexed using priority scheduling in order to flexibly share resources and increase the overall control performance. By separating stabilization and performance optimization, and providing each NCS with only a minimal amount of guaranteed real-time network resources, we can circumvent the scalability issues of our previous approach.

With our approach, opportunistic NCS traffic can be treated at arbitrary service levels, all the way down to best effort—at the cost of reduced control performance—as our approach is designed specifically without making any assumptions about the QoS for those transmissions. However, for the sake of simplicity, we assume in

most of our analysis that opportunistic NCS traffic up to a certain capacity is also transported using TSN, where excess opportunistic transmissions are dropped according to the priority scheduling policy. This assumption allows us to (a) derive conditions for optimizing the control performance, and (b) draw a fair comparison to previous work in our evaluation.

In summary, this chapter contains the following contributions:

- A mixed-criticality communication service for heterogeneous[1] NCS.
- A corresponding nominal NCS model that allows guaranteed stability analysis, worst-case performance analysis, and worst-case optimal controller design using standard control theoretic methods, when using our communication service.
- A performance-optimizing packet priority scheduling policy for the opportunistic portion of the traffic.
- A pessimistic performance model and corresponding controller design conditions that guarantee stability and preclude performance degradation even if opportunistic traffic is treated arbitrarily (i.e., unscheduled).

The remainder of this chapter is organized as follows. First, we give a brief review of related work in Section 4.2. In Section 4.3 we present our system model, consisting of an application-layer control system model, a transmission model for deterministic and opportunistic traffic, and a model for the packet scheduler. In Section 4.4 we formalize our problem of packet prioritization subject to stability guarantees and performance optimization. We then derive a suitable packet prioritization scheme in Section 4.5 and conduct a numerical simulation study in Section 4.6 to evaluate the effectiveness of our approach and compare it to our previous work. Finally, we conclude this chapter with a short discussion and outlook in Section 4.7.

4.2 Related Work

In [BA15] the suitability of weakly hard real-time scheduling for NCS has been investigated by deriving sufficient stability conditions. In [Zha+08], an online heuristic is developed for optimizing the H_∞ performance of a set of control systems by adapting their task periods subject to rate monotonic processor scheduling. In [Bar+17], the effect of skipped control jobs on the stabilizability of NCS is formalized using a weaker asymptotic condition and a stronger condition with a

[1] Each NCS can have different system dynamics and architecture, i.e., location of the controller in the network, as described in Section 4.3.1.

limited burst length, and appropriate task scheduling algorithms developed. The performance-optimal scheduling of control tasks on multiprocessor platforms under partitioned earliest deadline first (EDF) scheduling is investigated in [RAZ16], which could also have implications for communication scheduling. State-based sampling has been investigated, e.g., in [RSJ13; MH14; FK15; GRP16], where each sensor uses local scheduling (i.e., periodic event triggering) policies to reduce the number of transmissions, while the aggregate traffic of all NCS is managed using probabilistic contention-based medium access, which introduces random packet loss. In [Mam+18], the queueing delay in a network shared by a set of NCS is controlled by designing appropriate event-triggering thresholds, which provide almost sure mean-square stability. By contrast, our work proposes to leverage *packet scheduling in the network*, rather than random dropping or source-based policing, to improve performance.

A dynamic state-based scheduling approach has been studied in [AGL15], where the scheduler switches between a set of event-based NCS for which a suboptimal controller is co-designed. Our own state-based scheduler presented in Chapter 3 determines a similar scheduling policy at lower computational cost and generalizes the transmission model to allow more than one NCS to transmit simultaneously at each sampling time.

A communication model similar to the one presented here has been developed in [Mar+10] for Controller Area Network (CAN) based NCS, where periodic *mandatory* control jobs are augmented with non-periodic *optional* control jobs to improve performance, and CAN-specific message encoding issues are discussed. While that approach is designed for single-input, single-output (SISO) systems and uses a control gain design based on so-called *accelerable control tasks* [BSS08] to prevent performance degradation through optional control jobs, our approach supports multiple-input, multiple-output (MIMO) systems with different architectural set-ups and provides controller design methods to maximize the performance benefit of each opportunistic transmission.

4.3 System Model

Consider a set of N different NCS as shown in Figure 4.1. The sensor nodes send datagrams to the actuator nodes over a shared communication medium, which is modelled as a queueing system with two traffic classes. Depending on the location of the controller, the payload of a datagram is either a *measurement* of the plant state generated by the sensor for the controller, or a *control input* generated by the controller for the actuator.[2]

[2] In Chapter 3, we only considered the former configuration.

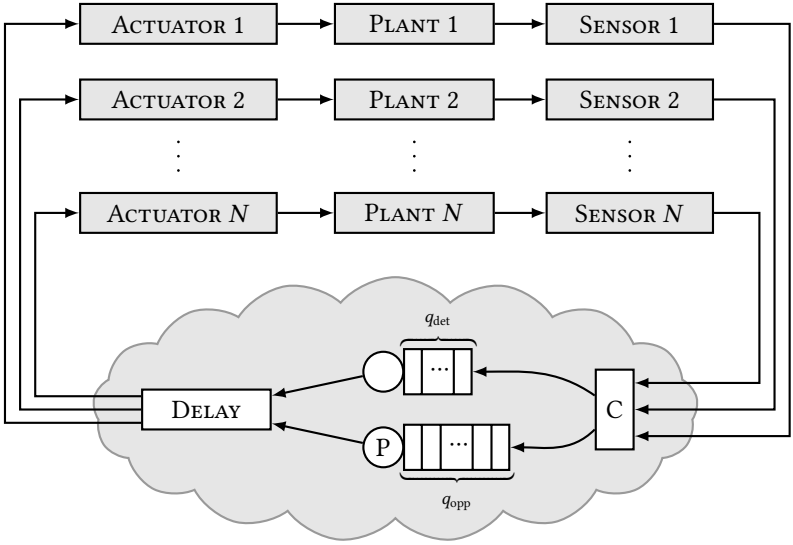


Figure 4.1: System architecture: N distinct control systems communicating over shared network.

Moreover, all datagrams are tagged at the source with a binary traffic classification (opportunistic or deterministic) and with a scalar priority value, as detailed in Section 4.3.2 and Section 4.3.3. The underlying network is modelled as a FIFO queue with capacity q_{det} for deterministic traffic and a separate priority queue (served in the order of the datagrams' associated scalar priority values) with capacity q_{opp} for opportunistic traffic.

4.3.1 Control System Model

As in Chapter 3, we assume that all NCS are sampled synchronously. The plant of NCS $i \in \{1, \dots, N\}$ is modelled as a discrete-time LTI system

$$x_{k+1}^i = A^i x_k^i + B^i u_k^i, \tag{4.1}$$

where $x_k^i \in \mathbb{R}^{n_i}$ is the state and $u_k^i \in \mathbb{R}^{m_i}$ is the applied input of plant i at time t_k. We assume that all (A^i, B^i) are controllable. Again, the QoC of system i is defined using the infinite-horizon LQ cost

$$J^i = \sum_{k=1}^{\infty} x_k^{i\top} Q^i x_k^i + 2 x_k^{i\top} H^i u_k^i + u_k^{i\top} R^i u_k^i, \tag{4.2}$$

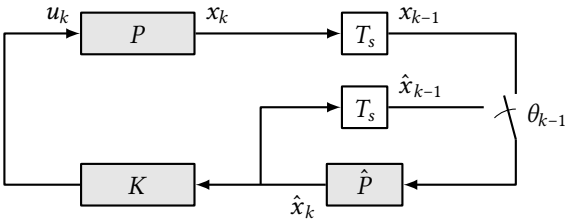


Figure 4.2: An individual NCS consisting of plant P, state predictor $\hat{P}$, and *actuator-colocated* controller K

where $\begin{bmatrix} Q^i & H^i \\ H^{i\top} & R^i \end{bmatrix} \succ 0$.

To model the control input u^i, we distinguish between two cases based on where the controller is located.

Actuator-colocated controller When located at the actuator node, as considered, e.g., in [BA12] and shown in Figure 4.2, the controller calculates the input based on a state estimate $\hat{x}_k^i \in \mathbb{R}^{n_i}$ that is updated based on the received state measurements

$$\hat{x}_k^i = \theta_{k-1}^i A^i x_{k-1}^i + \left(1 - \theta_{k-1}^i\right) A^i \hat{x}_{k-1}^i + B^i u_{k-1}^i, \tag{4.3}$$

$$u_k^i = -K^i \hat{x}_k^i, \tag{4.4}$$

where θ_k^i indicates whether the datagram transmitted at time t_k, which contains the state measurement x_k^i of NCS i, has been successfully received at the controller by time t_{k+1} ($\theta = 1$) or not ($\theta = 0$). The state estimate update (4.3) compensates for a constant transfer delay of T_s and for dropped packets. We assume that a controller gain matrix K^i is given such that $A^i - B^i K^i$ is Schur. For our evaluations we use a standard LQR optimal controller for K^i, which satisfies this condition under the assumptions given above.

Sensor-colocated controller When located at the sensor node, as considered, e.g., in [LA17] and shown in Figure 4.3, the controller transmits a predictive control input

$$\hat{u}_{k+1}^i = -K_k^i \left(A^i x_k^i + B^i u_k^i\right) \tag{4.5}$$

at time t_k. Note that we allow for time-variant control matrices K_k^i in this set-up. Design conditions for these matrices will be discussed in Section 4.5. At time t_{k+1}, the actuator applies a control input according to the policy

$$u_{k+1}^i = \theta_k^i \hat{u}_{k+1}^i + \left(1 - \theta_k^i\right) h^i u_k^i, \tag{4.6}$$

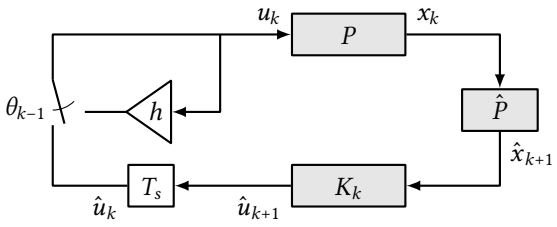


Figure 4.3: An individual NCS consisting of plant P, state predictor $\hat{P}$, and (time-varying) *sensor-colocated* controller K_k

where the right hand term is used for loss compensation. Again, θ_k^i indicates if the datagram transmitted at time t_k, which in this case contains $\hat{u}_{k+1}^i$, was successfully received at the actuator by t_{k+1}. The hold parameter $h^i \in \{0, 1\}$ determines whether to use the *zero* or the *hold* strategy for packet loss compensation, namely, if the control input should be reset to zero or rather be maintained at the previous value when no new datagram arrives. A discussion of these two strategies in the context of random packet loss can be found in [Sch09].

4.3.2 Transmission Model

In this chapter, we adopt the DOTS model presented in Section 2.2 for decoupling stability guarantees and performance optimization. Thereby, we distinguish between deterministic (guaranteed) and opportunistic transmissions. For this, we perform a time-based classification of datagrams into these two traffic classes for each control loop.

We assume that there is an underlying grid of timeslots for all transmissions which is determined by the sampling times t_k. For each NCS i, we denote the phase shift as s^i and the deterministic transmission period as d^i. Thereby, the transmission times t_k of all deterministic transmissions of NCS i are characterized by

$$k \equiv s^i \pmod{d^i}, \tag{4.7}$$

and we denote the set of all NCS with a deterministic transmission at time t_k as $D_k = \{i \mid k \equiv s^i \pmod{d^i}\} \subseteq \{1, \ldots, N\}$. We assume that deterministic datagrams are marked accordingly, for instance using the Ethernet PCP header field specified in IEEE 802.1Q, and

$$i \in D_k \implies \theta_k^i = 1, \tag{4.8}$$

as the corresponding transmissions are guaranteed. All other transmission slots are opportunistic, and carry no *a priori* guarantees.

Deterministic transmissions are handled as (isolated) real-time traffic with strictly bounded queueing delay. We therefore assume that $\left|D_k\right| \leq q_{\text{det}}$, i.e., the total number of deterministic transmissions in each timeslot is no greater than the capacity for deterministic transmissions, which corresponds to the size of a batch of datagrams that can be transferred with an overall end-to-end transfer delay of at most one sampling period T_s.

Note that the network model implied by these assumptions, as shown in Figure 4.1, need not be realized using a single switch, but can be considered as an abstraction of a multi-hop network with features such as those provided by standard TSN, specifically IEEE 802.1Qbv. While these standards define the necessary mechanisms for realizing highly synchronized TDMA schedules on switches, a number of algorithms have been proposed in the literature [Cra+16; DN16; NDR16] for calculating these schedules for a set of (periodic) time-triggered traffic flows, such that deterministic end-to-end delay bounds are guaranteed. Note also that transmission times t_k need not be strictly synchronized among all NCS (which simply implies worst-case queueing delay), but may also be shifted on time scales smaller that T_s, depending on the chosen TSN scheduling approach. Finally, q_{det} is not required to be constant, but can be taken as time-varying without loss of generality.

4.3.3 Packet Priority Scheduler Model

While we assume that—from the perspective of an individual application—no assumptions can be made about the success of an opportunistic transmission in isolation, we introduce a fixed-capacity priority queueing model in order to reason about θ_k^i, $i \notin D_k$ over the *set* of all NCS for the purpose of formulating a joint performance optimization goal in Section 4.4.[3]

As shown in Figure 4.1, we assume that all datagrams classified as opportunistic traffic are added to a queue with capacity q_{opp}. Furthermore, any opportunistic datagram from NCS i in slot k is tagged with a scalar packet priority v_k^i at the source, which we will specify in Section 4.5. The datagrams in the opportunistic queue are served by descending priority, so that that the q_{opp} highest-priority datagrams are delivered within (at most) one sampling period T_s, whereas the lower-priority datagrams are considered to be dropped.

Therefore, if we sort all opportunistic datagrams in time slot k according to

[3] Note, however, that our core contributions are still valid even when no guarantees whatsoever are made for opportunistic transmissions. Most importantly, stability guarantees are completely independent from the service level for opportunistic slots.

their packet priorities

$$v_k^{i_1} \geq v_k^{i_2} \geq \cdots \geq v_k^{i_{N-|D_k|}}, \quad i_j \notin D_k, \tag{4.9}$$

$$\text{then} \quad \theta_k^i = \begin{cases} 1 & i \in \{i_1, i_2, \ldots, i_{q_{\text{opp}}}\} \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

This is equivalent to choosing a subset of q_{opp} datagrams for which the sum of packet priority values is maximal, i.e.

$$\sum_{i \notin D_k} \theta_k^i v_k^i = \max_{\substack{|S| \leq q_{\text{opp}} \\ S \cap D_k = \emptyset}} \sum_{i \in S} v_k^i. \tag{4.11}$$

Note that, like for deterministic traffic, q_{opp} is not required to be constant. For instance, if a constant capacity for q real-time datagrams is available in each slot k through a corresponding (static) TSN schedule, of which q_{det}^k deterministic transmissions are allotted, the "leftover" capacity $q_{\text{opp}}^k = q - q_{\text{det}}^k$ could be made available for opportunistic transmissions (e.g., using the same TSN schedule with PCP 7 for deterministic and PCP 6 for opportunistic datagrams).

We stress that the (continuous) packet priority v_k^i—which defines the order of datagrams *within* the (abstract) priority queue—does *not* correspond to the the 3-bit PCP value in the IEEE 802.1Q header, which is used to assign datagrams to FIFO ordered switch-internal queues with *different* priorities. While priority queueing is not readily available in commodity hardware, it can be implemented, e.g., as a virtual network function (VNF) [LC15] in software. In Section 3.7.1, we have already shown a corresponding proof-of-concept middlebox implementation of a high-performance priority queue scheduler using DPDK. Network function virtualization (NFV) platforms such as NetVM [HRW15] or ClickOS [Mar+14] allow easy deployment and high-throughput realization of VNFs. In [Siv+13], it has even been suggested to directly control the fast-path scheduling and queueing behaviour of high performance hardware switches in a "software-defined" manner. Moreover, we point out that the conceptual priority queue in our model can be realized by an equivalent cascade of (identical) priority queues in a multi-hop network, since the priority order is transitive and queueing delays are only incurred at bottleneck points.

4.4 Problem Statement

Our goal is to provide a communication service for a set of NCS based on the system model presented in Section 4.3, which satisfies the following conditions:

(C1) Stability and worst-case LQ cost analysis can be performed at the application layer for each NCS i individually, depending only on its deterministic slot period d^i and not requiring any knowledge about opportunistic transmissions.

(C2) A controller minimizing the worst-case LQ cost (i.e., minimum QoC) can be determined at the application layer for each NCS i individually, depending only on its deterministic slot period d^i and not requiring any knowledge about opportunistic transmissions.

(C3) The packet priorities for all opportunistic datagrams are chosen such that the overall LQ cost

$$\mathcal{J} = \sum_{i=1}^{N} J^i \tag{4.12}$$

is minimal, subject to the constraint that at any time t_k no assumptions are made about any opportunistic transmissions at times $t > t_k$.

4.5 Opportunistic Packet Prioritization

In Section 4.5.1, we first derive a *nominal* model of each individual control system that expresses the evolution of the plant state from one deterministic slot to the next, under the assumption that no opportunistic transmissions are successful. Based on this, we derive a pessimistic cost-to-go model and stability condition for (C1) and develop surrogate models in the same form as (4.1)–(4.2) for the controller design to satisfy (C2). In Section 4.5.2, we derive packet priorities and additional constraints for satisfying condition (C3).

In the following, we perform most of our analysis on a single representative NCS in isolation. Therefore, we will drop the superscript index i identifying a specific NCS wherever it is not technically necessary. Note that, nonetheless, all NCS need not be identical but can have different dynamics and controller configurations.

4.5.1 Nominal Application Model

Without loss of generality, we assume that the NCS under observation has phase shift $s = 0$ for deterministic transmissions, as we can simply apply the transformation $k \mapsto k + s$ to get back to the reference transmission grid. Therefore, the deterministic transmission slots are given by $k_l = ld$, $l \in \mathbb{N}$ and the dynamics from k_l to k_{l+1} for an initial state x_{k_l} are given by applying (4.1), together with (4.3)–(4.4) for the actuator-colocated or (4.5)–(4.6) for the sensor-colocated controller, recursively with $\theta_{k_l} = 1$ and $\theta_{k_l+1} = \cdots = \theta_{k_l+d-1} = 0$.

Augmented state space To express this more conveniently, we first use the familiar approach of rewriting the system equations in terms of an augmented state variable χ, given by either

$$\chi_k = \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} \quad \text{or} \quad \chi_k = \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{4.13}$$

for the actuator-colocated or sensor-colocated controller, respectively. The system equations and LQ cost in terms of the augmented state are

$$\chi_{k+1} = \mathcal{T}_k^{\theta_k} \cdot \mathcal{A} \cdot \chi_k \tag{4.14}$$

$$J = \sum_{k=1}^{\infty} \chi_k^\top \mathcal{Q} \chi_k, \tag{4.15}$$

with

$$\mathcal{A} = \begin{bmatrix} A & -BK \\ 0 & A - BK \end{bmatrix}, \ \mathcal{T}_k = \begin{bmatrix} I & 0 \\ I & 0 \end{bmatrix}, \ \mathcal{Q} = \begin{bmatrix} Q & -HK \\ -K^\top H^\top & K^\top R K \end{bmatrix} \tag{4.16}$$

for the actuator-colocated controller and

$$\mathcal{A} = \begin{bmatrix} A & B \\ 0 & hI \end{bmatrix}, \ \mathcal{T}_k = \begin{bmatrix} I & 0 \\ -K_k & 0 \end{bmatrix}, \ \mathcal{Q} = \begin{bmatrix} Q & H \\ H^\top & R \end{bmatrix} \tag{4.17}$$

for the sensor-colocated controller. Note that $\mathcal{T}_k^1 = \mathcal{T}_k$ and $\mathcal{T}_k^0 = I$, meaning that we describe each iteration step of the NCS model in the augmented state space as an open-loop iteration, i.e., $\mathcal{A}\chi_k$, possibly followed by a multiplication from the left with $\mathcal{T}_k$ which emulates the update of $\hat{x}_{k+1}$ or u_{k+1} as a result of a successful transmission if $\theta_k = 1$.

Nominal model The system model for the nominal execution, where only deterministic transmissions are successful, can now be written in the augmented state space as

$$\chi_{k_{l+1}} = \Phi_d \cdot \chi_{k_l} := \mathcal{A}^{d-1} \mathcal{T} \mathcal{A} \cdot \chi_{k_l} \tag{4.18}$$

and the LQ cost incurred during each (d-step) iteration of the nominal model is

$$\sum_{k=k_l}^{k_{l+1}-1} \chi_k^\top \mathcal{Q} \chi_k = \chi_{k_l}^\top \mathcal{Q} \chi_{k_l} + \sum_{j=0}^{d-2} \chi_{k_l}^\top \mathcal{A}^\top \mathcal{T}^\top \mathcal{A}^{j\top} \mathcal{Q} \mathcal{A}^j \mathcal{T} \mathcal{A} \chi_{k_l} \tag{4.19}$$

$$= \chi_{k_l}^\top \left(\mathcal{Q} + \mathcal{A}^\top \mathcal{T}^\top \Omega_{d-1} \mathcal{T} \mathcal{A} \right) \chi_{k_l}, \tag{4.20}$$

where we define

$$\Omega_0 = 0, \quad \Omega_{n+1} = \mathcal{Q} + \mathcal{A}^\top \Omega_n \mathcal{A} \tag{4.21}$$

for convenience of notation. Note that the nominal model is sampled only at the deterministic slots, as the dynamics in the opportunistic slots is completely predetermined in the nominal case. It allows us to determine the stability and nominal (i.e., worst-case) LQ cost of the NCS, as shown in the following Theorem.

Theorem 4.1 (Nominal LQ Cost and Stability)**.** *The LQ cost* (4.15) *of the nominal model* (4.18) *in terms of the augmented state space for an initial value χ_0 in the initial deterministic slot k_0 is given by*

$$J_{nom} = \chi_0^\top P_{nom} \chi_0, \tag{4.22}$$

where P_{nom} solves the discrete Lyapunov equation

$$P_{nom} = \Phi_d^\top P_{nom} \Phi_d + Q_{nom} \tag{4.23}$$

$$\text{with } Q_{nom} = \mathcal{Q} + \mathcal{A}^\top \mathcal{T}^\top \Omega_{d-1} \mathcal{T} \mathcal{A}. \tag{4.24}$$

Moreover, if Q_{nom} and P_{nom} are positive definite, the nominal system is globally asymptotically stable.

Proof. Given the linear model (4.18) for the nominal system and associated stage cost (4.19)–(4.20), we use the *ansatz*

$$J_{\text{nom}}\left(\chi_{k_l}\right) = \chi_{k_l}^\top P_{\text{nom}} \chi_{k_l} \tag{4.25}$$

for the nominal cost-to-go at any deterministic slot χ_{k_l}, which we can then write recursively as

$$\begin{aligned} J_{\text{nom}}\left(\chi_{k_l}\right) &= \chi_{k_l}^\top \left(\mathcal{Q} + \mathcal{A}^\top \mathcal{T}^\top \Omega_{d-1} \mathcal{T} \mathcal{A}\right) \chi_{k_l} + J_{\text{nom}}\left(\chi_{k_{l+1}}\right) \\ &= \chi_{k_l}^\top \left(\mathcal{Q} + \mathcal{A}^\top \mathcal{T}^\top \Omega_{d-1} \mathcal{T} \mathcal{A} + \Phi_d^\top P_{\text{nom}} \Phi_d\right) \chi_{k_l}. \end{aligned} \tag{4.26}$$

The identity of (4.25) and (4.26) leads directly to (4.23)–(4.24). The stability result follows by noting that under the given conditions $J_{\text{nom}}(\chi)$ is also a Lyapunov function for the nominal system (cf., e.g., [Kha02]). ■

Of course, standard numerical methods are available for solving the Lyapunov equation (4.23), e.g., [Lau79]. Note that these results are derived under the assumptions that all opportunistic transmissions are unsuccessful. However, we will show in Section 4.5.2 that stability of the nominal system also implies stability of realizations with opportunistic transmissions, and that the nominal LQ cost is not exceeded, if certain constraints are imposed on opportunistic transmissions.

Controller design The previously described stability and cost analysis for the nominal case does not depend on a certain controller gain K, which may be arbitrarily chosen. Note that the matrices in (4.16)–(4.17) depend on the chosen (possibly time-varying) controller gain. Therefore, we cannot simply derive the nominal optimal controller using a standard LQR design for the nominal system. Rather, to satisfy (C2) we present Theorem 4.2 and Theorem 4.3 for designing controllers that minimize the nominal LQ cost.

Designing an optimal controller for the nominal system is simplest in the actuator-colocated setup. Due to the dynamic control law (4.3)–(4.4), the dynamics of the open-loop system coincides with that of the closed-loop system after the first deterministic slot, whereafter $\hat{x}_k \equiv x_k$. This leads directly to the following Theorem.

Theorem 4.2 (Optimal Actuator-colocated Controller)**.** *Let $\hat{K}$ be the gain matrix of the LQ optimal controller $u_k = -\hat{K}x_k$ for the original open-loop system* (4.1)*, with the LQ stage cost as in* (4.2)*. Setting $K = \hat{K}$ in* (4.4) *also minimizes the LQ cost of the nominal system.*

For the sensor-colocated setup, on the other hand, the control value transmitted in deterministic slot k' is either held or set to zero after the following slot until the next deterministic slot $k' + d$. Therefore, to design an optimal controller for the nominal system, we must take into account both the state evolution and the accumulated LQ stage cost over the complete interval between consecutive deterministic slots. As the control gain $K_{k'}$ from deterministic slot k' is applied predictively to obtain $u_{k'+1} = -K_{k'}x_{k'+1}$, cf. (4.5), we shift the frame of reference for the nominal model to the dynamics from step $k' + 1$ to $k' + d + 1$. (Note that this does not impact the total LQ cost, as the stage cost up to and including the first deterministic slot is unaffected by the choice of $K_{k'}$.) By repeatedly applying (4.14) we get

$$\chi_{k'+d+1} = \begin{bmatrix} x_{k'+d+1} \\ u_{k'+d+1} \end{bmatrix} = \mathcal{T}_{k'+d}\mathcal{A}^d \begin{bmatrix} x_{k'+1} \\ u_{k'+1} \end{bmatrix}. \tag{4.27}$$

Furthermore, the accumulated LQ stage cost over this interval is

$$\sum_{k=k'+1}^{k'+d} \chi_k^\top \mathcal{Q}\chi_k = \sum_{j=0}^{d-1} \chi_{k'+1}^\top \mathcal{A}^{j\top}\mathcal{Q}\mathcal{A}^j\chi_{k'+1} = \chi_{k'+1}^\top \Omega_d \chi_{k'+1}. \tag{4.28}$$

Theorem 4.3 (Optimal Sensor-colocated Controller)**.** *Let $\hat{K}$ be the gain matrix of*

the LQ optimal controller $u_k = -\hat{K}x_k$ *for the surrogate system*

$$x_{k+1} = A_d \cdot x_k + B_d \cdot u_k, \tag{4.29}$$

$$J = \sum_{k=0}^{\infty} x_k^\top Q_d x_k + 2x_k^\top H_d u_k + u_k^\top R_d u_k, \tag{4.30}$$

$$\textit{where} \begin{bmatrix} A_d & B_d \\ * & * \end{bmatrix} = \mathcal{A}^d \textit{ and } \begin{bmatrix} Q_d & H_d \\ H_d^\top & R_d \end{bmatrix} = \Omega_d. \tag{4.31}$$

Setting $K_{k_l} = \hat{K}$ *in* (4.5) *for deterministic slots* k_l *also minimizes the LQ cost of the nominal system.*

Proof. The surrogate model (4.29)–(4.30) with (4.31) is obtained directly by rewriting (4.27)–(4.28) in terms of $x_{k'+d+1}$, $x_{k'+1}$, and $u_{k'+1}$ from the original state space, and substituting the time index $k + j$ for $k' + jd + 1$. ■

Of course, the surrogate system models in Theorem 4.2 and Theorem 4.3 can also be used to design non-optimal controllers with other desired characteristics.

4.5.2 Opportunistic Performance Optimization

Now that we have derived means to analyse the stability and performance of each NCS individually, and to design controllers under nominal (worst-case) assumptions about the communication service, we are ready to tackle condition (C3), i.e., to leverage packet priority scheduling of the opportunistic traffic to optimize the overall control performance of all NCS. As the priority scheduler in the network maximizes in each time step k the sum of packet priorities of successful transmissions, cf. (4.11), the natural approach would be to assign to each datagram the potential *reduction in LQ cost-to-go* if its transmission were successful as a packet priority, i.e., in terms of the augmented state

$$v_k = J\big(\mathcal{A}\chi_k\big) - J\big(\mathcal{T}\mathcal{A}\chi_k\big) \tag{4.32}$$

Thereby, the total LQ cost-to-go of all NCS would be directly minimized.

Unfortunately, the cost-to-go function $J(\chi)$ in (4.32) admits no known closed-form expression as far as we know, because it depends on all future opportunistic transmissions, which in turn would depend on the packet priorities v_k defined by (4.32). Therefore, we pessimistically approximate the cost-to-go for an opportunistic slot k' by assuming that all future opportunistic transmissions would fail, i.e., $\theta_k^i \equiv 0$, $k > k'$, $i \notin D_k$. This pessimistic cost-to-go is given by the accumulated stage cost from slot k up to the next deterministic slot, plus the nominal cost-to-go J_{nom} at that time. For the sake of convenience, we consider only opportunistic slots

in the interval $k = 1, \dots, d-1$ between the first two deterministic slots, noting as in Section 4.5.1 that the results can be generalized to all time steps by appropriate transformation of k.

We generalize the nominal cost-to-go to opportunistic slots by defining $J_k^\theta(\chi_k)$ as the pessimistic cost-to-go as described above where $\theta_k = \theta$ indicates the (potential) success of the opportunistic transmission in slot k:

$$J_k^\theta(\chi_k) := \chi_k^\top P_k^\theta \chi_k \tag{4.33}$$

$$= \sum_{j=k}^{d-1} \chi_j^\top \mathcal{Q} \chi_j + J_{\text{nom}}(\chi_d) \tag{4.34}$$

$$= \chi_k^\top \left(\mathcal{Q} + \sum_{j=0}^{d-k-2} \mathcal{A}^\top \mathcal{T}_k^{\theta\top} \mathcal{A}^{j\top} \mathcal{Q} \mathcal{A}^j \mathcal{T}_k^\theta \mathcal{A} \right) \chi_k \tag{4.35}$$

$$+ J_{\text{nom}}\left(\mathcal{A}^{d-k-1} \mathcal{T}_k^\theta \mathcal{A} \chi_k \right)$$

$$= \chi_k^\top \left(\mathcal{Q} + \mathcal{A}^\top \mathcal{T}_k^{\theta\top} \Big(\underbrace{\Omega_{d\text{-}k\text{-}1} + \mathcal{A}^{d\text{-}k\text{-}1\top} P_{\text{nom}} \mathcal{A}^{d\text{-}k\text{-}1}}_{=: M_k} \Big) \mathcal{T}_k^\theta \mathcal{A} \right) \chi_k \tag{4.36}$$

Lemma 4.4 (LQ Cost With Opportunistic Transmissions)**.** *Given the following priority metric*

$$v_k := \chi_k \Delta_k \chi_k, \tag{4.37}$$

$$\text{where } \Delta_k = \mathcal{A}^\top \left(M_k - \mathcal{T}_k^\top M_k \mathcal{T}_k \right) \mathcal{A} \tag{4.38}$$

$$\text{and } M_k = \Omega_{d-k-1} + \mathcal{A}^{d-k-1\top} P_{nom} \mathcal{A}^{d-k-1} \tag{4.39}$$

for any opportunistic slot $0 < k < d$ *(w.l.o.g.), the actual LQ cost, i.e., the cost-to-go from an (initial) deterministic slot* $k = 0$*, of an NCS is*

$$J = J(\chi_0) = J_{nom}(\chi_0) - v_k \tag{4.40}$$

if exactly one opportunistic transmission is successful in slot k *or more generally*

$$J = J(\chi_0) = J_{nom}(\chi_0) - \sum_{k=1}^{d-1} \theta_k v_k \tag{4.41}$$

if all θ_k *were known for* $0 < k < d$*. This result generalizes to all slots by substituting* $(k \bmod d)$ *for* k *in* (4.38)–(4.39) *and* $\left\lfloor \frac{k}{d} \right\rfloor \cdot d$ *for* 0 *in* (4.40)–(4.41)*.*

Proof. From (4.33)–(4.36) and (4.39), the nominal cost decrease for opportunistic transmissions (4.32) follows as

$$J_k^0(\chi_k) - J_k^1(\chi_k) = \chi_k^\top \left(P_k^0 - P_k^1\right)\chi_k, \tag{4.42}$$

$$= \chi_k^\top \left(\mathcal{Q} + \mathcal{A}^\top M_k \mathcal{A} - \mathcal{Q} - \mathcal{A}^\top \mathcal{T}_k^\top M_k \mathcal{T}_k \mathcal{A}\right)\chi_k \tag{4.43}$$

$$= \chi_k^\top \mathcal{A}^\top \left(M_k - \mathcal{T}_k^\top M_k \mathcal{T}_k\right)\mathcal{A}\chi_k = v_k. \tag{4.44}$$

The results (4.40)–(4.41) follow directly from the definition of J_k^θ. ■

Note that the cost in (4.40) is no greater than the nominal cost J_{nom} only if the priority metric v_k is non-negative. While this is satisfied for the actuator-colocated controller if K is chosen LQ optimal, it cannot be guaranteed for the sensor-colocated controller due to the zero or hold compensation scheme, if no special assumptions about the gain matrices K_k are made. Therefore, we can only guarantee that the cost is not increased by suppressing transmissions with $v_k < 0$ similarly as in [Mar+10], which leads directly to the following Theorem.

Theorem 4.5 (Opportunistically LQ-optimal Packet Priorities)**.** *Let no assumptions be available at time k about $\theta_{k'}$ for any opportunistic slots $k' > k$. Using the priority scheduler* (4.9)–(4.10) *for the opportunistic traffic of a set of N NCS, the total LQ cost* (4.12) *is minimized by choosing the packet priorities in slot k as in* (4.37)[4] *with the additional constraint*

$$v_k^i < 0 \implies \theta_k^i = 0. \tag{4.45}$$

Proof. The result follows directly from the priority scheduler definition (4.11) and the joint LQ cost definition (4.12), taking the individual LQ costs J^i as in (4.40). ■

Just as in Theorem 4.1, we can also infer the stability of each NCS from its associated (pessimistic) cost-to-go function.

Theorem 4.6 (Stability With Opportunistic Transmissions)**.** *If the nominal system* (4.18) *is globally asymptotically stable, then the NCS with (arbitrary) opportunistic transmissions subject to the constraint* (4.45) *is also globally asymptotically stable.*

Proof. We use the Lyapunov function J_{nom} for the nominal system known from the proof of Theorem 4.1 as a Lyapunov function candidate for arbitrary realizations with opportunistic transmissions. From (4.41) we can derive that for any deterministic slot k', the remaining nominal cost-to-go in the following step is

$$J(\chi_{k'+1}) = J(\chi_{k'}) - \chi_{k'}^\top \mathcal{Q}\chi_{k'} = J_{\text{nom}}(\chi_{k'}) - \chi_{k'}^\top \mathcal{Q}\chi_{k'} - \sum_{k=k'+1}^{k'+d-1} \theta_k v_k$$

[4]Note that the matrices $\mathcal{A}$, $\mathcal{T}$, and M_k are of course different for each NCS in general.

which is an upper bound for the value $J_{\text{nom}}(\chi_{k'+d})$ in the next deterministic slot. From (4.45), we know that the sum over $\theta_k v_k$ is non-negative, which implies that $J_{\text{nom}}(\chi_{k'+d}) \leq J(\chi_{k'+1}) < J_{\text{nom}}(\chi_{k'})$. Therefore, J_{nom} remains a Lyapunov function for the nominal system—sampled at the deterministic slots—when opportunistic transmissions are introduced. Because the term $\chi_k^\top P_{\text{nom}} \chi_k$ for each opportunistic slot $k = k'+1, \dots, k'+d-1$ is uniformly bounded by $|\lambda_{\max}|^{2d} J_{\text{nom}}\left(\chi_{k'}\right)$, where $\lambda_{\max}$ is the maximum absolute eigenvalue of $\mathcal{T}_k^\theta \mathcal{A}$ over k and θ, it follows that the behaviour for the opportunistic slots is also asymptotically stable, cf. [Clo+06]. ■

To further optimize the performance of the sensor-colocated controller for opportunistic transmissions, we can design K_k such that the nominal cost decrease, and thereby the packet priorities, v_k become maximal. This is the result of our final Theorem.

Theorem 4.7 (Optimal Opportunistic Controller)**.** *Under the assumptions of Theorem 4.5, the time-varying gain matrix for the sensor-collocated controller in each opportunistic slot minimizing the total LQ cost* (4.12) *is given by*

$$K_k = \left(M_k^{uu}\right)^{-1} M_k^{ux} \tag{4.46}$$

$$\text{where } \begin{bmatrix} M_k^{xx} & M_k^{ux\top} \\ M_k^{ux} & M_k^{uu} \end{bmatrix} = M_k. \tag{4.47}$$

Proof. Because $J_k^0(\chi_k)$ is independent of K_k, v_k can be maximized by minimizing $J_k^1(\chi_k)$ with respect to K_k. First, we recall from (4.33)–(4.36) that

$$J_k^1\left(\chi_k\right) = \chi_k^\top \left(\mathcal{Q} + \mathcal{A}^\top \mathcal{T}_k^\top M_k \mathcal{T}_k \mathcal{A} \right) \chi_k \tag{4.48}$$

$$= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \mathcal{Q} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix}^\top M_k \begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix}. \tag{4.49}$$

The first term is independent of K_k. Therefore, the whole expression can be minimized by determining the extremum of the second term with respect to u_{k+1} (noting that $u_{k+1} = -K_k x_{k+1}$, cf. eqs. (4.1), (4.5), (4.6) with $\theta_k = 1$). Using the fact that M_k is symmetric (due to $\mathcal{Q} \succ 0$ and $P_{\text{nom}} \succ 0$), and partitioning it as in (4.47), the minimum of $J_k^1\left(\chi_k\right)$ with respect to u_{k+1} is determined by

$$\frac{\partial}{\partial u_{k+1}} \chi_{k+1}^\top M_k \chi_{k+1} = 2M_k^{ux} x_{k+1} + 2M_k^{uu} u_{k+1} \stackrel{!}{=} 0 \tag{4.50}$$

$$\implies u_{k+1} = -\left(M_k^{uu}\right)^{-1} M_k^{ux} x_{k+1} \tag{4.51}$$

which equals $-K_k x_{k+1}$ as in (4.46). ■

4.6 Numerical Evaluation

In order to demonstrate the effectiveness of our approach and compare it to our previous work, we now present some simulation-based evaluation results. To allow scaling the problem size in a meaningful way, we ran simulations of N identical NCS with the plant model

$$x_{k+1}^i = \begin{bmatrix} 0 & 1 \\ 0.4 & 0.7 \end{bmatrix} x_k^i + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k^i, \tag{4.52}$$

which has poles at −0.468115 and 1.06811. Each plant has a different initial state defined by $x_0^i = \frac{i}{N} \cdot \begin{bmatrix} 1 & 1 \end{bmatrix}^\top$, $i = 1, \dots, N$. We assume that a total capacity of $q = q_{\text{det}} + q_{\text{opp}} = 2$ is available. For evaluating our approach we divide this equally between deterministic and opportunistic transmissions, i.e., $q_{\text{det}} = q_{\text{opp}} = 1$, and NCS i has deterministic phase shift $s^i = i$ and period $d^i = N$, such that deterministic slots are allocated to all NCS in a round-robin fashion. Our simulation experiments begin at $k = 1$. Note that this leads to a pessimistic realization, since the magnitude of each NCS's initial value is proportional to the time it has to wait for its first deterministic transmission.

In our evaluation, we compare the scalability and performance of the approach presented in this chapter (denoted as *opp-prio* in the figures) by comparing it to the following alternative approaches:

- Random scheduling of opportunistic transmissions, i.e., using random packet priorities (denoted as *opp-rand* in the figures, using mean values over 100 realizations).
- Round-robin scheduling, where we set $q_{\text{det}} = 2$ and $q_{\text{opp}} = 0$, and the deterministic slot periods are changed to $d^i = \left\lceil \frac{N}{2} \right\rceil$ (denoted as *RR* in the figures).
- State-dependent priority scheduling without deterministic slots, where we set $q_{\text{det}} = 0$ and $q_{\text{opp}} = 2$, and all transmissions are scheduled by packet priority determined as in Chapter 3 (denoted as *prio* in the figures).[5]

Scalability First, we compare the number N of NCS that can be accommodated by the communication service in our approach and in our previous work. In the introduction we already mentioned that the scalability of the scheduler in Chapter 3 is limited by the feasibility of the associated SDP optimization problem.

[5] For the sake of comparison, we extended the scheduler in Chapter 3 to also support the sensor-colocated controller setup, by exchanging the augmented state models in (3.9)–(3.10) with corresponding models using the expressions from (4.17).

Table 4.1: Maximum number $N_{\max}$ of NCS depending on controller set-up (*AC* for actuator-, *SC* for sensor-colocated), loss compensation strategy (*zero* or *hold*) and approach used.

	AC	SC (zero)	SC (hold)
$N_{\max}$ for opportunistic scheduler	1044	460	526
$N_{\max}$ for scheduler in Chapter 3	17	11	5

By contrast, in our current approach, the number of NCS is limited by the schedulability of their deterministic slots. For our evaluation set-up described above, we can accommodate $N \leq q \cdot d_{\max}$ NCS, where $d_{\max}$ is the maximum period for which the nominal system is stable, assuming that we allocate all of the capacity $q = q_{\text{det}}$ to deterministic slots. The comparison chart in Table 4.1 shows that our current approach vastly outperforms our previous work in terms of schedulability for all considered controller configurations.

Next, we evaluate the time required for calculating the packet priority coefficient matrices Δ_k^i, $i = 1, \dots, N$, $k = 1, \dots, d^i - 1$ for a set of N heterogeneous NCS with the same state dimensions as (4.52), where we consider only the actuator-colocated set-up, since it provides the largest range of feasible solutions for the scheduler in Chapter 3. Figure 4.4 shows the average runtimes using one core on a PC with 2.7 GHz Intel Core i7-2620M CPU and 16 GB RAM. In Chapter 3, the coupling of stabilization and performance optimization requires solving an SDP optimization problem, for which we used the commercial Mosek solver [MOS18]. As our current approach only requires the solution of the Lyapunov equation (4.23) and evaluation of the matrix expressions (4.38)–(4.39), it can be seen to be faster by a factor of about 300. Note also that incrementally adding an NCS typically requires only the calculation of additional matrices Δ_{d+1}^i, which based on the previously calculated matrices takes on the order of one millisecond, whereas in Chapter 3 we need to solve a complete SDP problem again.

Performance In Figure 4.5, we compare the control performance achieved for a simulation of the set-up with NCS (4.52) over 1000 time steps using our approach in comparison with the alternative approaches listed above. We plot the total LQ cost J relative to the nominal (i.e., worst-case) cost $J_{\text{nom}} = J_k^0(\chi_1)$, over varying total number N of NCS. We can see that our approach always outperforms the random and round-robin approaches. For the actuator-colocated set-up, the performance of our approach is inferior to the performance achieved by the scheduler in Chapter 3, which is offset however by the vastly improved scalability. For the sensor-colocated set-up with zero loss compensation ($h = 0$), the performance

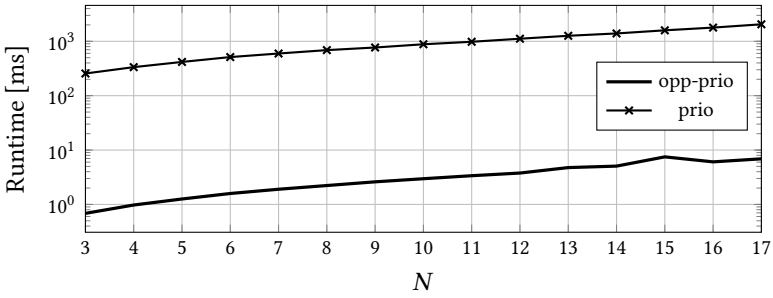


Figure 4.4: Runtime for calculating $\{\Delta_k^i\}$ for a set of N heterogeneous NCS with the same dimensions as (4.52) using the opportunistic scheduler (opp-prio), compared to the scheduler in Chapter 3 (prio).

of our approach is comparable to that of our previous work. Finally, for the sensor-colocated set-up with hold loss compensation ($h = 1$), our approach is superior, which is also due to the overall bad performance of our previous work in this set-up.

4.7 Summary and Discussion

In this chapter, we have introduced a mixed-criticality communication service leveraging the DOTS model, which allows us to guarantee stability for individual NCS and optimize the joint control performance for a set of heterogeneous NCS. Our priority scheduler allows opportunistic traffic to be handled with arbitrary QoS without impacting the stability or worst-case performance provided by deterministic transmissions, provided that the appropriate conditions are met (Theorem 4.6 and/or Theorem 4.7).

We have provided a nominal NCS model that allows guaranteed stability and worst-case performance analysis (Theorem 4.1) and worst-case optimal controller design (Theorems 4.2 and 4.3) using standard control theoretic methods. For opportunistic transmissions, a performance-optimizing packet priority scheduling policy (Theorem 4.5) and controller design method (Theorem 4.7) have been derived.

An interesting consequence of our priority scheduler analysis is that the coefficient matrices Δ_k for the calculation of packet priorities depend only on the location of the corresponding opportunistic slot relative to the next/previous deterministic slot. This enables a static analysis of the maximum possible datagram priority, e.g., by determining the spectral radii of $\mathcal{T}_k^\top \mathcal{A}_k^\top \Delta_k \mathcal{A}_k \mathcal{T}_k$ for $k = 1, \dots, d-1$

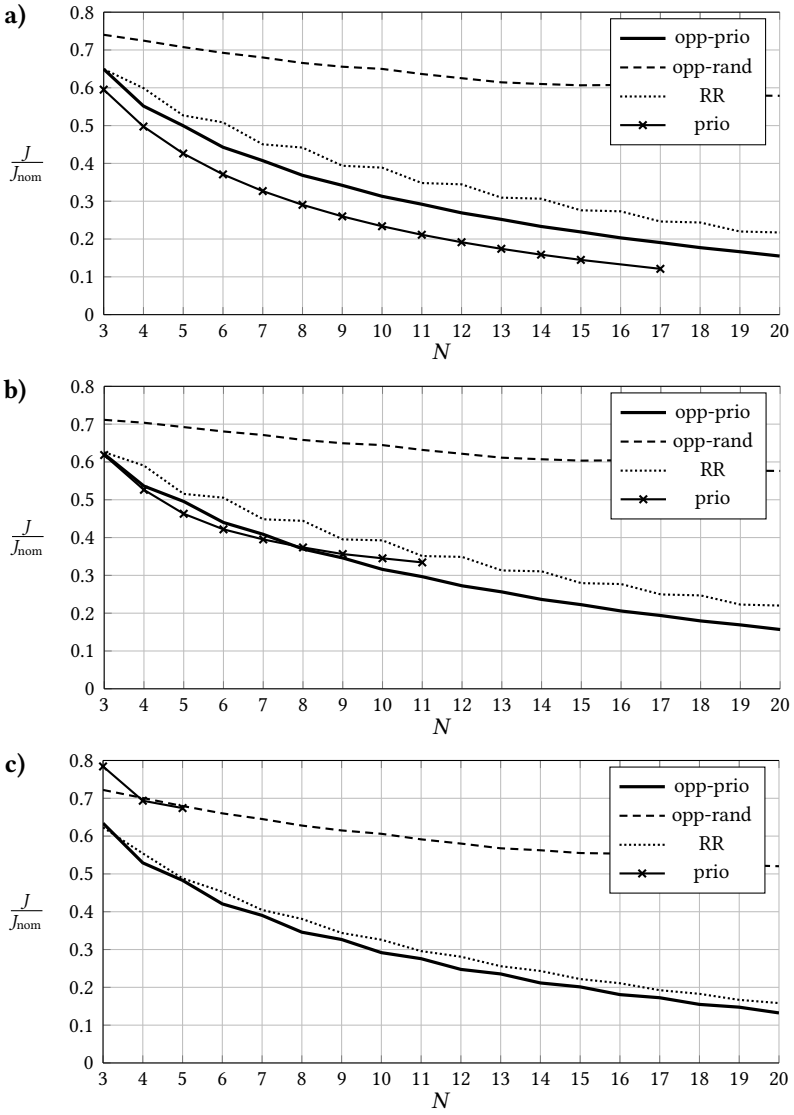


Figure 4.5: LQ cost comparison of opportunistic priority scheduler (opp-prio) with random (opp-rand), round-robin (RR), and priority scheduling as in Chapter 3 (prio);
a) *actuator-colocated* setup,
b) *sensor-colocated* setup with *zero* strategy,
c) *sensor-colocated* setup with *hold* strategy.

to get a profile of the expected packet priorities of the opportunistic transmissions relative to the norm of the state in the previous deterministic slot. This would open up avenues for future work, such as optimizing the scheduling of deterministic slots (i.e., phase shifts s^i and possibly periods d^i) in order to minimize the overall expected priorities in each opportunistic slots, thereby minimizing the number of collisions of high-priority—and therefore high-performance-gain—datagrams. Similarly, the routing of flows for opportunistic transmissions could be optimized by choosing disjoint paths to minimize high-priority collisions. Recent results on so-called *complemental flows* [Fal+19a] address scheduling and routing mechanisms for such types of traffic, and could benefit from application-specific knowledge about the utility of individual transmission slots. Finally, our nominal model and associated results could be generalized to allow *non-equidistant* deterministic slots in order to improve the schedulability of deterministic transmissions.

5 Routing

5.1 Introduction

In Chapters 3 and 4, we considered the scheduling problem for multiple control systems under the assumption that real-time transmission can be guaranteed at least for a certain limited bandwidth of NCS traffic, either for an entire traffic class or even for some transmissions individual systems. Currently, this assumption largely limits the applicability of those results to LANs. In fact, control systems are conventionally implemented using field-bus networks with deterministic real-time guarantees at the data link layer, such as the well-known CAN and PROFIBUS and more specialized Ethernets such as EtherCAT and PROFINET [GH13]. As already discussed in the previous chapters, TSN [IEE16] now offers mechanisms for supporting real-time communication in standard IEEE 802.3 Ethernet. In wide-area networks, however, implementing deterministic control models based on a best-effort service as provided by the current Internet is unfeasible. While communication services with guaranteed delay bounds, such as IntServ, would be an appropriate base for building Internet-scale deterministic NCS, we cannot expect them to be widely available due to lack of scalability [WM03]. Standardization efforts by the IETF for deterministic networking (DetNet) [Fin+18] are still ongoing.

As a consequence, we focus our attention in this chapter on probabilistic control models, which only require probabilistic delay guarantees. The control methods for probabilistic communication proposed in the literature (e.g., [BA11; BA13; BA14; Hee+10; QSG07]) incorporate mathematical stochastic network models to enable the rigorous analysis of the (stochastic) QoC depending on network parameters. However, they do not address the problem of mapping these models to contemporary communication services. Of course, it is not straightforward to determine which is better: an unreliable service with very short delays, or a more reliable service with larger delays. In [BA13], the trade-off between latency and *in-time* arrival probability for the optimal control of LTI systems has been investigated in the context of increasing communication reliability using retransmissions. We spin this idea further in order to explore this trade-off for optimal routing.

In this chapter, we propose a novel communication service tailored to the spe-

cific needs of control systems based on simple probabilistic network models. This transport-layer service, called *NC-service*, enables connections from the sensor to the controller of an NCS. When requesting a connection, the control application specifies an application-specific QoS model, which is based on the closed-loop LQ cost. This model specifies the required in-time message arrival probability as a function of the acceptable delay (given by the sampling period). Due to the range of admissible combinations of sampling period and arrival probability, there are possibly a multitude of network paths that fulfil the application's performance requirement. We can leverage this degree of freedom by selecting the path that incurs the minimal network load. This renders the NC-service a cross-layer service by incorporating routing functionality.

Selecting the optimal path based on the performance metric described above requires a customized routing mechanism. We can utilize state of the art SDN technologies to implement this mechanism, making our approach applicable to standard networking infrastructures. This approach is well suited to demonstrate the core benefits of SDN, namely, the flexibility to add new network control logic, ease of implementing a logically centralized network service, and integration of the application and network into a holistic system.

In detail, this chapter contains the following contributions:

- An end-to-end transport abstraction for NCS connections based on a probabilistic QoS model tailored to the stochastic control model that expresses a lower bound on QoC.
- The formal specification of a constrained routing problem to optimally utilize network bandwidth resources subject to QoC constraints.
- A logically centralized routing algorithm, which can be implemented using an SDN-based architecture.
- An evaluation demonstrating the effectiveness and feasibility of our approach.

The rest of this chapter is organized as follows. In Section 5.2 we discuss related work. We introduce our system model in Section 5.3 and propose an SDN-based architecture for the NC-service in Section 5.4. After deriving the QoC requirements for a class of LQR control systems, we describe the transport layer functionality in Section 5.5. We then show an efficient implementation of this service on the network layer by formulating and solving an optimal routing problem in Section 5.6. Section 5.7 evaluating our approach with a simulation study, and is followed by a discussion and concluding summary in Section 5.9.

5.2 Related Work

The stabilizability of NCS depending on the network delay distribution has been investigated in [BA14] employing a network model similar to ours. There, it is shown that the minimum arrival probability can be determined analytically under certain conditions, however, not regarding the LQ performance. In [Xu+14], a method for designing controllers and sampling periods is presented, depending on the delay distribution not of the network but of the task response times induced by priority scheduling of multiple controllers on a shared processor.

In [NDR16], the provisioning of real-time (low-latency, low-jitter) communication for time-triggered traffic using so-called time-sensitive software-defined networking (TSSDN) is considered as a routing and transmission scheduling problem. The TSSDN approach is suitable for LANs with SDN capable commodity switches that do not provide TSN mechanisms for time-aware scheduling at the data link layer. However, the approach is restricted to networks that are under the control of one management domain, since the routing algorithm requires global knowledge of all time-sensitive flows in the network, for which the highest-priority traffic class must be reserved exclusively. In contrast, our approach considers the routing of individual flows and only requires knowledge of the latency distributions in the network.

The shortest path problem with on-time arrival reliability is considered in [NW09] in the context of travel times in transportation networks. However, it addresses only the delay minimization with respect to a fixed on-time arrival probability[1] and the dual, i.e., maximizing the on-time arrival probability with respect to a fixed delay. In contrast, neither delay or arrival probability are fixed in our approach, but are coupled through the QoC constraint, which allows us to consider all paths that are admissible for achieving a fixed closed-loop performance.

5.3 System Model

Our system model is shown in Figure 5.1. The sensor node is connected to the controller over a network, which we assume to be a packet-switched IP network with best-effort service semantics. State measurement vectors from the sensor node may be dropped with variable loss probability and experience a variable delay. For the sake of simplicity, we assume that the actuators are co-located with the controller. Therefore, our focus in this chapter is on the connection between the sensor node and the controller.

[1] Specifically, "the latest possible departure time and the associated route to attain a given probability of arriving at the destination at a specified arrival time or earlier" [NW09]

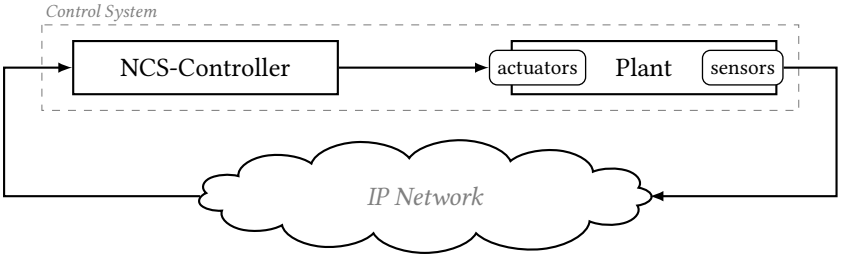


Figure 5.1: System model with networked sensor–controller connection

For the packet-switched network, we assume the availability of standard multi-layer switches that can forward flows based on IP addresses and higher layer information such as port numbers. Moreover, we assume that these switches support the OpenFlow standard [ONF15] for configuring forwarding tables and enabling our SDN-based architecture described next. Thereby, we confine our investigation to NCS within one autonomous system. Mechanisms for inter-domain networking are beyond the scope of this thesis.

5.3.1 Control System Model

In this chapter, we use an NCS model from [BA13]. For the communication between sensor and controller, this model assumes a *constant delay* T_s and *arrival probability* p_a. Furthermore, losses are assumed to be independent and identically distributed (i.i.d.). The continuous-time plant is modelled using the ODE

$$\dot{x}(t) = A_c x(t) + B_c u(t) + w(t), \tag{5.1}$$

where $x \in \mathbb{R}^n$ is the state vector of the plant, which is measured by the sensor, $u \in \mathbb{R}^m$ is the input vector of the plant, and $w \sim \mathcal{N}(0, W_c)$ is an external random Gaussian disturbance.

The sensor is sampled periodically at time instants t_k, where $t_{k+1} = t_k + T_s$. The sampling period T_s is assumed to be equal to the constant network delay in the network model. However, in this chapter we consider T_s an adjustable parameter which simultaneously determines the transmission rate of the sensor and specifies the delay bound for determining the in-time arrival probability.

To compensate for the network delay and possible message loss, we employ the same one-step predictive controller as in Section 3.3.1. This results in an

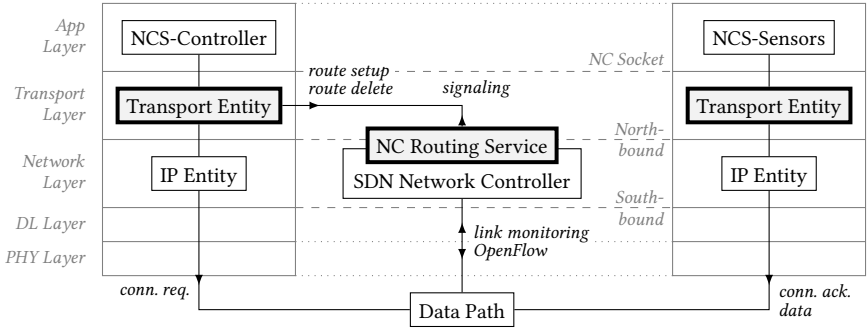


Figure 5.2: Service architecture: NC-service components with bold borders

equivalent discrete-time model for the closed-loop system

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{5.2}$$

$$\hat{x}_{k+1} = \theta_k Ax_k + (1 - \theta_k)A\hat{x}_k + Bu_k, \tag{5.3}$$

$$u_k = -K\hat{x}_k, \tag{5.4}$$

where θ_k is a binary indicator for the in-time arrival of measurements. It is now important to note that the system matrices A and B as well as the noise covariance matrix W for $w_k \sim \mathcal{N}(0, W)$ and the controller gain matrix K depend on the sampling period T_s. For the discretization of the continuous-time plant model (5.1), we refer to [BA13, Sec. III], and for the design of the optimal controller to [BA13, Sec. V]. Subject to the constraints imposed by the network model, the controller K is designed such as to minimize the LQ cost functional

$$J = \lim_{T\to\infty} \frac{1}{T} \int_0^T x^\top(t)Qx(t) + u^\top(t)Ru(t)\,\mathrm{d}t. \tag{5.5}$$

As both the external disturbances experienced by the plant and the packet loss incurred by the network are stochastic, the LQ cost J is also a random variable. However, the *expected value* of J under optimal control can be determined numerically as a function of the sampling period T_s and the arrival probability p_a, as shown in [BA13, Theorem 2]. In the following, we denote this function for the expected LQ cost as $\mathrm{E}[J] = \mathrm{E}_J\left(T_s, p_a\right)$.

5.4 Service Architecture

The service architecture of our NC-service is depicted in Figure 5.2, which shows the relevant components in the five-layer Internet model. As previously men-

tioned, our proposed NC-service provides a transport layer interface to the control application and utilizes a custom routing mechanism to implement the route of the corresponding NC-connection at the network layer.

The transport functionality of the NC-service is implemented by protocol entities on the end-systems hosting the sensors and controller. We denote the interface between application and NC-service as an *NC-socket* similar to classic sockets for datagram and streaming services. The transport layer service is responsible for setting up an *NC-connection* between sensor and controller, tearing down connections, data transfer, QoS arbitration, and signalling changes in network conditions to the application as detailed in the following sections.

In order to set up an NC-connection with a given QoS, a path with certain loss and delay properties has to be established, which might be adapted during the life-time of the connection due to changing network conditions. The set-up and maintenance of routes implementing the required QoS with minimum network resources is the task of the *NC routing service.* According to the SDN paradigm, we implement this service using a logically centralized *SDN network controller* which exposes the routing service to the transport layer through its northbound interface. To calculate a route, the NC routing service obtains necessary QoS information from the transport layer entity and gathers a global view onto the network through monitoring of the link properties over the southbound SDN interface (e.g., using counters to detect lost packets or active probing to determine delays). Based on this global view, the NC routing service implements the optimal route in the network by installing the necessary flow table entries in the OpenFlow-enabled switches (based on the IP addresses and port numbers of NCS controller process and sensor process and the protocol ID of the NC transport protocol). Whenever the routing service adapts the route or identifies the need to adjust application parameters in order to maintain QoS, these changes are signalled back to the transport layer. Note that route adaptations can be performed in a non-disruptive fashion without packet loss[2] using techniques for consistent network updates [Rei+12]. Data packets follow the usual data path through the protocol stack.

5.5 NC Transport Service

We now proceed to design a suitable transport abstraction for NCS which will be provided by the NC Transport Service (NCT-service for short). In particular, this abstraction includes the QoS definition used to set up NC-connections. On the one hand, this definition has to provide the network with the information needed to efficiently implement NC-connections on the network layer, i.e., to determine appropriate network paths. On the other hand, the QoS parameters must be in

[2] although jitter will increase most likely

line with the performance metrics of the application itself. Before presenting the QoS concept of the proposed NCT-service, we therefore revisit the underlying NCS model.

5.5.1 QoS Specification

We can use the NCS model to map application performance parameters to network QoS parameters. From an application perspective, the control engineer wishes the NCS to maintain a certain performance. She therefore provides an upper bound $J_{\max}$ for the application's expected cost:

$$\mathrm{E}_J\left(T_s, p_a\right) \leq J_{\max}. \tag{5.6}$$

Given this constraint, we can specify the QoS for the NCS as the required minimum arrival probability given T_s:

Definition 5.1 (Minimum Arrival Probability for NCS)**.** The *minimum arrival probability* function of an NCS with upper cost bound $J_{\max}$ is given by

$$p_{\min}\left(T_s\right) = \inf\left\{\, p_a \mid \mathrm{E}_J\left(T_s, p_a\right) \leq J_{\max} \,\right\}. \tag{5.7}$$

The function $p_{\min}$ defines a meaningful relationship between delay and arrival probability as parameters of the NCS model regarding the application's performance. However, to develop a QoS specification for NC-connections, we must also consider the constraints imposed on these parameters by the network. In the following, we describe how the arrival probability p_a depends on the delay bound T_s.

Recall that in the control model described above the packet delivery time of state measurements from sensor to controller is assumed to be constant. From the application's perspective, this assumption is perfectly justified since sensing and actuation only occur at predefined time instants. However, in the network we must account for varying delays. State measurements experiencing a delay of less than one sampling period T_s can be buffered by the NCT-service until the next sampling instant. Those experiencing a greater delay can be discarded as, at their arrival time, the controller has already applied some input u based on its state estimate. Effectively, the application parameter p_a describes the probability of timely end-to-end delivery of NCS packets within the delay bound T_s. We can use this *effective arrival probability* to characterize an NC-connection:

Definition 5.2 (Effective Arrival Probability)**.** Let σ denote an end-to-end NC-connection from sensor to controller which imposes a random delay τ^σ and

random packet loss with probability p_{loss}^{σ}. The effective arrival probability $p_{\sigma}(t)$ of this connection is defined as

$$p_{\sigma}(t) = P\left(\tau^{\sigma} \leq t\right) \cdot \left(1 - p_{\text{loss}}^{\sigma}\right), \tag{5.8}$$

with respect to the delay bound t.

Thus, an NC-connection satisfies the quality of service requirements of the NCS if we can find a finite positive sampling time T_s such that the effective arrival probability exceeds the required minimum arrival probability:

$$\exists_{T_s > T_{\min}} \quad p_{\sigma}(T_s) \geq p_{\min}(T_s) \tag{5.9}$$

Here, we also consider a lower bound $T_{\min}$ on the acceptable minimum sampling period as a system parameter to limit the data rate of NCS traffic. Note that expressing QoS as a function of the application-layer parameter T_s is beneficial since it allows the network to choose an optimal sampling period based on the current network delay and loss such that the NCS requirements are fulfilled with minimal induced network load as we will present in Section 5.6. By using this lightweight but expressive interface, we retain all degrees of freedom (T_s and network path), while providing separation of concerns between the control application and the network.

5.5.2 NCT Service Interface

As outlined in Section 5.4, we propose a slight extension of the classic datagram socket as an application interface for the NCT-service, which we denote as NC-socket. Apart from opening and closing NC-connections and sending and receiving sensor data, NC-sockets also support the passing of NCS-specific QoS parameters to the NCT-service and signalling of events between network and NCS, e.g., when the QoS cannot be fulfilled anymore. In the following, we briefly describe the NC-socket interface as well as the transport service semantics.

To set up an NC-connection, the controller (which we assume initiates the connection) must pass the QoS specification function $p_{\min}(\cdot)$ defined in (5.7) as an additional parameter to the NC-socket `connect` primitive. The QoS specification is passed on via the northbound interface of the SDN network controller to the NC routing service (cf. below) which determines the optimal route and sampling period T_s. The transport entities return the chosen sampling period T_s to the controller and sensor nodes through the NC-socket `connect`/`accept` primitives, respectively, and signal handlers are provided to notify the NCS application if the sampling time has to be changed in order to maintain QoC.

Both connection end-points are also provided with a reference time t_0. Once the NC-connection is established, the sensor node starts sending measurements

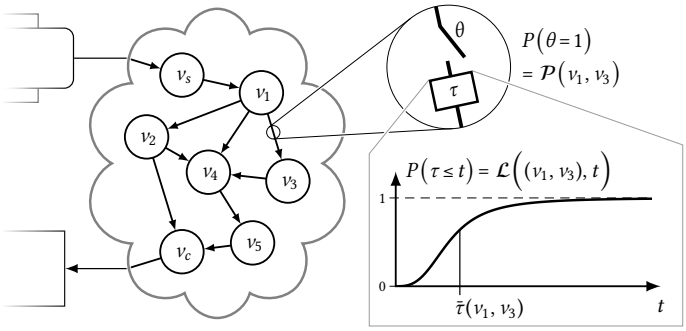


Figure 5.3: Network model for NC-flow implementation. Each link $e \in E$ obeys a distinct random packet loss process θ with arrival probability $\mathcal{P}(e)$, and a distinct random delay process τ with latency distribution $\mathcal{L}(e, \cdot)$.

to the controller at sampling times $t_k = t_0 + k \cdot T_s$. On each send call, the transport entity includes a sample number $k = \left\lfloor \frac{t - t_0}{T_s} \right\rfloor$ corresponding to the current sampling time t_k when the measurement was taken. Violations of the delay bound T_s can be detected at the controller using the time of arrival, t_0, and k. Thus, the clocks of the controller and the corresponding sensor node need to be synchronized using a standard clock synchronization technique, such as NTP, PTP, or GPS. If no valid measurement is received within a sampling period, `receive` times out and the controller performs a state prediction as shown in Section 5.3.1.

5.6 NC Routing Service

Based on the QoS definition of NC-connections, we can now focus on the problem of finding *optimal* network paths for forwarding packets of an NC-connection (or for short NC-flows) such that the requested QoS is fulfilled and the network load is minimized. To this end, we define a formal routing problem for NC-flows in this section after first introducing the prerequisites for this formulation, namely, a formal network model and definitions of the latency and loss properties of network paths as well as network load. Finally, we present an optimal routing algorithm for NC-flows.

5.6.1 Network Model

We model the communication network as a directed graph $G = \left(V, E, \mathcal{P}, \mathcal{L}\right)$ as depicted in Figure 5.3. It comprises a set V of vertices modelling network nodes

(multi-layer switches or hosts), and a set $E \subseteq V \times V$ of directed edges modelling network links between nodes. G contains two special nodes: v_s representing the sensor node, and v_c representing the host of the controller; all other nodes are (multilayer) switches. The map $\mathcal{P} : E \longrightarrow [0, 1]$ describes the arrival probability of each link $e \in E$, and $\mathcal{L} : E \times \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ represents the cumulative distribution function (CDF) of each link's latency. As notational shorthand for the corresponding probability density function (PDF), we write $\ell(e, t)$, i.e.,

$$\mathcal{L}(e, t) = \int_0^t \ell(e, \tau)\, \mathrm{d}\tau.$$

The transmission of NCS packets is modelled by a stochastic process $\{\theta_k^e, \tau_k^e\}_{k \in \mathbb{N}}$ for each link $e \in E$. Here, $\theta_k^e \in \{0, 1\}$ is a random binary variable representing packet arrival, i.e., $\theta_k^e = 0$ if and only if the measurement $x(t_k)$ is dropped on link e, and $\tau_k^e \in \mathbb{R}^+$ denotes the random network delay of $x(t_k)$ on link e, assuming it was not lost. As stated above, the stochastic properties of this process are defined by G:

$$P\left(\theta_k^e = 1\right) = \mathcal{P}(e) \qquad \text{(arrival probability)} \tag{5.10}$$

$$P\left(\tau_k^e \leq t\right) = \mathcal{L}(e, t), \ t \in \mathbb{R}^+ \qquad \text{(latency distribution)} \tag{5.11}$$

In accordance with the control model, we assume that the process $\{\theta_k^e, \tau_k^e\}$ is i.i.d. between sampling instants t_k and independent over all links. Further, let the mean latency of link $e \in E$ be denoted by $\bar{\tau}(e) = \mathrm{E}[\tau_k^e]$.

In order to obtain the properties of NC-flow routes as link sequences, we define a k-hop path as a tuple of the visited nodes $\sigma = \left(v^1, v^2, \ldots, v^{k+1}\right) \in V^{k+1}$. For $k \geq 1$, a path must be connected in G and thus satisfy $\left(v^i, v^{i+1}\right) \in E$ for all $i = 1, \ldots, k$. For $k = 0$ the path reduces to a single node, which we refer to a as a *degenerate path*. We say that two paths $\sigma_1 = \left(v_1^1, v_1^2, \ldots, v_1^m\right)$ and $\sigma_2 = \left(v_2^1, v_2^2, \ldots, v_2^n\right)$ are *adjacent* if the terminal node of the first coincides with the initial node of the second, i.e., $v_1^m = v_2^1$. The *concatenation* of two adjacent paths is denoted as $\sigma_1 \circ \sigma_2 = \left(v_1^1, v_1^2, \ldots, v_1^m, v_2^2, \ldots, v_2^n\right)$. Finally, we say that two distinct paths are *parallel* if the initial and terminal nodes of both paths coincide, i.e., $v_1^1 = v_2^1$ and $v_1^m = v_2^n$.

5.6.2 Path Properties

By regarding the stochastic properties of link concatenation with respect to delay and loss, $\mathcal{P}$ and $\mathcal{L}$ can also be extended to paths. For the sake of consistency, we consider any degenerate path $\sigma \in V$ as a perfectly reliable and instantaneous link, i.e., $\mathcal{P}(\sigma) = 1$, $\mathcal{L}(\sigma, t) \equiv 1$, and $\bar{\tau}(\sigma) = 0$.

Proposition 5.1. *Given two adjacent paths σ_1 and σ_2, the overall arrival probability and delay probability density of the composite path $\sigma_1 \circ \sigma_2$ are given by*

$$\mathcal{P}\big(\sigma_1 \circ \sigma_2\big) = \mathcal{P}(\sigma_1) \cdot \mathcal{P}(\sigma_2) \tag{5.12}$$

$$\text{and } \ell\big(\sigma_1 \circ \sigma_2, t\big) = \ell(\sigma_1, t) * \ell(\sigma_2, t), \tag{5.13}$$

where $*$ denotes convolution with respect to t.

Proof. First, we assume that one of the paths, say σ_2 without loss of generality, is degenerate. Then σ_1 must remain invariant under concatenation, i.e. $\sigma_1 \circ \sigma_2 = \sigma_1$. This is clearly fulfilled by (5.12) and (5.13), noting that $\mathcal{P}(\sigma_2) = 1$ and $\ell(\sigma_2, t) = \delta(t)$ is the Dirac delta distribution:

$$\mathcal{P}\big(\sigma_1 \circ \sigma_2\big) = \mathcal{P}(\sigma_1) \cdot 1 = \mathcal{P}(\sigma_1)$$
$$\ell\big(\sigma_1 \circ \sigma_2, t\big) = \ell(\sigma_1, \tau) * \delta(t) = \ell(\sigma_1, t)$$

Now assume that σ_1 and σ_2 are two adjacent links. The joint arrival probability $\mathcal{P}\big(\sigma_1 \circ \sigma_2\big)$ is then given by

$$P\big(\theta_1 = 1 \wedge \theta_2 = 1\big) = \mathcal{P}(\sigma_1) \cdot \mathcal{P}(\sigma_2),$$

thus (5.12) holds. Moreover, the random latency $\tau^{\sigma_1 \circ \sigma_2}$ of the composite path is the sum of the latencies of the constituent paths $\tau^{\sigma_1} + \tau^{\sigma_2}$. Therefore, the probability density function of the former is given by the convolution of the PDFs of the two latter random delay variables, and (5.13) holds. The remaining proof for the case when σ_1 and σ_2 are general adjacent paths follows by induction. ■

Note that we can express the effective arrival probability of NCS packets over a path σ in G as $p_\sigma(T_s) = \mathcal{P}(\sigma) \cdot \mathcal{L}(\sigma, T_s)$.

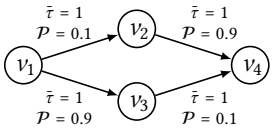
Obviously, the above proposition implies:

$$\mathcal{P}\big(\sigma_1 \circ \sigma_2\big) \leq \min\Big\{\mathcal{P}(\sigma_1), \mathcal{P}(\sigma_2)\Big\}$$
$$\text{and } \mathcal{L}\big(\sigma_1 \circ \sigma_2, t\big) \leq \min\Big\{\mathcal{L}(\sigma_1, t), \mathcal{L}(\sigma_2, t)\Big\},\ \forall t \in \mathbb{R}^+.$$

That is, the effective arrival probability of a composite path cannot be larger than those of its constituents.

5.6.3 Load Metric

Next, we introduce the network load metric used in our optimization criterion. Due to the stochastic nature of our link model, we use as a load measure the expected value $L = \mathrm{E}\big[b \cdot \tau^\sigma\big]$ of the bandwidth–delay product, where b is the



σ	$\bar{\tau}(\sigma)$	$\mathcal{P}(\sigma)$	$\mathrm{E}_\tau(\sigma)$
(v_1, v_2, v_4)	2	0.09	1.1
(v_1, v_3, v_4)	2	0.09	1.9

Figure 5.4: Expected transit time E_τ: although both paths from v_1 to v_4 have the same end-to-end arrival probability and mean delay, the path via v_2 incurs a shorter expected transit time and thus lower expected network load.

data rate of the traffic generated by the control application and τ^σ is the random latency of the path. To account for the fact that a packet lost somewhere along a route still incurs a certain amount of "wasted" network bandwidth, e.g., by occupying queue space, we use the following path metric to capture the expected delay experienced by any packet along a given path.

Definition 5.3 (Expected transit time)**.** The expected time that a packet spends in transit on path $\sigma = e_1 \circ e_2 \circ e_3 \circ \cdots$ in G is

$$\mathrm{E}_\tau(\sigma) \stackrel{\text{def}}{=} \left(\bar{\tau}(e_1) + \mathcal{P}(e_1) \cdot \left(\bar{\tau}(e_2) + \mathcal{P}(e_2) \cdot \left(\bar{\tau}(e_3) + \ldots \right) \right) \right).$$

Naturally, a path that is less reliable on the first hops incurs a shorter expected transit time than one that is less reliable on the last hops, and is thus favourable in terms of network load. Similar metrics have previously been investigated in the context of energy-efficient routing in wireless sensor networks, cf. [Sau+06]. Note that the expected transit time of any link $e \in E$ is simply its mean delay $\mathrm{E}_\tau(e) = \bar{\tau}(e)$ and that of a degenerate path $\sigma \in V$ is $\mathrm{E}_\tau(\sigma) = 0$. Also, given two adjacent paths σ_1 and σ_2, the expected transit time of the composite path $\sigma_1 \circ \sigma_2$ is

$$\mathrm{E}_\tau\big(\sigma_1 \circ \sigma_2\big) = \mathrm{E}_\tau(\sigma_1) + \mathcal{P}(\sigma_1)\,\mathrm{E}_\tau(\sigma_2), \tag{5.14}$$

which follows directly from Definition 5.3.

As the data rate of the NCS traffic is inversely proportional to the sampling period T_s, the induced network load on path σ with non-zero sampling period T_s can now be calculated as $L(\sigma, T_s) = \mathrm{E}_\tau(\sigma) \cdot T_s^{-1}$.

5.6.4 Routing Objective

Based on the formulation of the network load $L(\sigma, T_s)$, we can now define our constrained optimization problem. Recall that our QoS specification requires that

for some $T_s \geq T_{\min}$, the minimum arrival probability function $p_{\min}(T_s)$ of the NCS be no larger than $p_\sigma(T_s)$, which defines the probability of receiving a sample in time considering the loss and delay distribution of the path σ. Let us denote the set of sampling times where this condition is satisfied for a particular path σ as its *feasible sampling periods*

$$\mathcal{T}_{\text{feas}}(\sigma) = \left\{ t \geq T_{\min} \;\middle|\; p_\sigma(t) \geq p_{\min}(t) \right\}.$$

In order to incur as little network load as possible, our goal is to minimize the number of transmissions, i.e., the sampling rate, within the prescribed QoS constraints. This is achieved for any particular path by choosing the *maximal feasible sampling period* given by

$$T_{\max}(\sigma) = \sup \mathcal{T}_{\text{feas}}(\sigma)$$

Note that the expected transit time only depends on the choice of path, but not on the sampling period. Therefore, the load on any given path is minimized by choosing $T_s = T_{\max}(\sigma)$, and we denote this minimal path load as $\tilde{L}(\sigma) = L\big(\sigma, T_{\max}(\sigma)\big)$.

A low-quality path would therefore necessitate a high sampling rate in order to satisfy the QoS requirements. However, a path is unfeasible if its set of feasible sampling periods is empty, due to too large of a loss probability or delay, for which not even choosing the maximum sampling rate $T_{\min}^{-1}$ can compensate. In this case, the maximum feasible sampling period and consequently also the minimal path load are undefined.

Now we are ready to formulate an optimization problem for NC-flow routing, where we determine a path from the sensor node v_s to the controller node v_c with minimal network load:

$$\begin{aligned} \min_{\sigma \in G} \quad & \tilde{L}(\sigma) = \frac{\mathrm{E}_\tau(\sigma)}{T_{\max}(\sigma)} \\ \text{s.t.} \quad & \sigma_1 = v_s, \\ & \sigma_n = v_c, \\ & \mathcal{T}_{\text{feas}}(\sigma) \neq \emptyset. \end{aligned} \tag{5.15}$$

5.6.5 Routing Algorithm

Being a constrained shortest path problem, the considered routing problem (5.15) belongs to the class of *NP*-complete problems (cf. [Zie01]). Moreover, the QoS constraint (5.9) is not expressed in terms of a scalar constraint metric; therefore, traditional approximation methods for constrained shortest path problems are

not readily applicable. However, we employ a DP approach and show that its application is practicable on realistic topologies in Section 5.7.

Algorithm 1: Routing Algorithm

Data: $G = (V, E, \mathcal{P}, \mathcal{L})$ – network graph
$p_{\min}(\cdot)$ – minimum arrival probability function (QoC specification)
$v_s, v_c \in V$ – sensor and controller node
Result: Optimal route σ_{opt} solving (5.15)

1 **foreach** $v_n \in V$ **do**
2 $\quad R(v_n) \leftarrow \varnothing$; // $R(v_n)$ tracks candidate paths from v_n to v_c
3 **end**
4 $R(v_c) \leftarrow \{v_c\}$; // initialize terminal path set on v_c
5 $M \leftarrow \{v_c\}$; // mark v_c for relaxation of ingoing edges
6 **repeat**
7 $\quad$ **foreach** $v_n \in M$ **do**
8 $\quad\quad$ **foreach** v_i *with* $(v_i, v_n) \in E$ **do**
9 $\quad\quad\quad$ Relax(v_i, v_n)
10 $\quad\quad$ **end**
11 $\quad\quad$ $M \leftarrow M \setminus \{v_n\}$; // unmark node v_n
12 $\quad$ **end**
13 **until** $M = \varnothing$;
14 $\sigma_{\text{opt}} \leftarrow \arg\min_{\sigma \in R(v_s)} \left(\mathrm{E}_\tau(\sigma) / T_{\max}(\sigma) \right)$;

Our algorithm is shown in Algorithms 1 and 2. We maintain for each node v_i a set $R(v_i)$ of residual candidate paths to the target node v_c, which is initialized in Algorithm 1, lines 1–4 as empty for all nodes except the controller node, where it contains the degenerate path (v_c). We then successively generate simple paths from each node towards v_c by relaxation of each node's incoming edges in Algorithm 1, lines 5–13. Upon relaxing an edge (v_i, v_j) in Algorithm 2, we evaluate for each path $\sigma \in R(v_j)$ the extended path $\hat{\sigma} = v_i \sigma$.

As the set of simple paths to any node in G scales with $\mathcal{O}(|V|!)$ in the worst case, we would want to avoid generating all these paths. Due to the problem structure, we can unfortunately not select the "best" path at every step as would be possible in an unconstrained shortest path problem. We can, however, identify non-optimal sub-paths using a necessary optimality condition given in the following proposition.

Proposition 5.2. *Consider two parallel paths σ_1 and σ_2 from any node v_i to the*

Algorithm 2: Edge Relaxation Function

```
1  Function Relax(v_i, v_j)
2      foreach σ ∈ R(v_j) do
3          σ̂ ⟵ v_i σ ;                                   // new candidate path
4          if v_i ∈ σ or σ̂ ∈ R(v_i) then                 // loop/duplicate detection
5              continue
6          end
7          calculate E_τ(σ̂), P(σ̂), and L(σ̂, t);
8          if P(σ̂)L(σ̂, t) < p_min(t) ∀_{t>T_min} then     // σ̂ unfeasible
9              continue
10         end
11         if ∃ σ ∈ R(v_i) : σ ≫ σ̂ then                  // σ̂ dominated
12             continue
13         end
14         R(v_i) ⟵ R(v_i) \ {σ | σ̂ ≫ σ} ;               // purge paths dominated by σ̂
15         R(v_i) ⟵ R(v_i) ∪ {σ̂} ;                       // add candidate path σ̂ to R(v_i)
16         M ⟵ M ∪ {v_i} ;                               // mark v_i for relaxation of ingoing edges
17     end
18 end
```

controller node v_c. If these paths satisfy

$$\mathcal{P}(\sigma_1)\mathcal{L}(\sigma_1, t) < \mathcal{P}(\sigma_2)\mathcal{L}(\sigma_2, t), \quad \forall t, \tag{5.16}$$

$$\mathrm{E}_\tau(\sigma_1) > \mathrm{E}_\tau(\sigma_2), \tag{5.17}$$

then the solution σ_{opt} of the optimal routing problem (5.15) cannot *contain σ_1 as a sub-path.*

Proof. Assume that Proposition 5.2 is false and σ_{opt} does in fact contain σ_1, that is $\sigma_{\mathrm{opt}} = \sigma \circ \sigma_1$, and the optimal cost is given by $L(\sigma_{\mathrm{opt}})$. Now we consider the alternative path $\sigma_{\mathrm{alt}} = \sigma \circ \sigma_2$. It follows from (5.17) that

$$\underbrace{\mathrm{E}_\tau(\sigma) + \mathcal{P}(\sigma)\cdot\mathrm{E}_\tau(\sigma_1)}_{\mathrm{E}_\tau(\sigma_{\mathrm{opt}})} > \underbrace{\mathrm{E}_\tau(\sigma) + \mathcal{P}(\sigma)\cdot\mathrm{E}_\tau(\sigma_2)}_{\mathrm{E}_\tau(\sigma_{\mathrm{alt}})}, \tag{5.18}$$

i.e. the expected transit time of σ_{alt} is greater than that of σ_{opt}. Moreover, we can express the difference of the effective arrival probability functions of σ_{alt} and σ_{opt}

as

$$\begin{aligned} p_{\sigma_{\text{alt}}}(t) - p_{\sigma_{\text{opt}}}(t) &= \mathcal{P}(\sigma_{\text{alt}})\mathcal{L}(\sigma_{\text{alt}}, t) - \mathcal{P}(\sigma_{\text{opt}})\mathcal{L}(\sigma_{\text{opt}}, t) \\ &= \Big(\mathcal{P}(\sigma)\ell(\sigma, t)\Big) * \Big(\mathcal{P}(\sigma_2)\mathcal{L}(\sigma_2, t) - \mathcal{P}(\sigma_1)\mathcal{L}(\sigma_1, t)\Big), \end{aligned}$$

where we use the fact that convolution is distributive and associative with scalar multiplication. Due to (5.16), the above expression is non-negative for all t, whereby

$$\begin{aligned} & p_{\sigma_{\text{opt}}}(t) \leq p_{\sigma_{\text{alt}}}(t), \quad \forall t, \\ \Longrightarrow \quad & T_{\max}(\sigma_{\text{opt}}) \leq T_{\max}(\sigma_{\text{alt}}). \end{aligned} \tag{5.19}$$

It now follows from (5.18) and (5.19) that

$$\frac{\mathrm{E}_\tau(\sigma_{\text{opt}})}{T_{\max}(\sigma_{\text{opt}})} > \frac{\mathrm{E}_\tau(\sigma_{\text{alt}})}{T_{\max}(\sigma_{\text{alt}})}$$

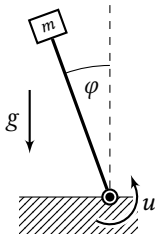
and therefore $L(\sigma_{\text{opt}}) > L(\sigma_{\text{alt}})$. However, this contradicts the assumption that σ_{opt} is the optimal path. ■

Based on this result, we can safely discard paths which meet the following condition as route candidates.

Definition 5.4. (Dominance) If two parallel paths σ_1 and σ_2 satisfy both (5.16) and (5.17), then we say that σ_2 *dominates* σ_1. As notational shorthand for this case, we write

$$\sigma_2 \gg \sigma_1.$$

Specifically, in Algorithm 2, lines 4–13, we discard the candidate $\hat{\sigma}$ if it contains a loop, is unfeasible, or dominated by any existing path in $R(v_i)$. Otherwise, we know that it is eligible as sub-path of the optimal solution, and we add it to $R(v_i)$. In line 14, we also purge $R(v_i)$ of all paths which are dominated by the newly added $\hat{\sigma}$, in order to further minimize the required number of relaxation steps. After a new candidate path was added to the set $R(v_i)$, we schedule in line 16 the incoming edges of v_i to be relaxed in the main loop. Thus, each $R(v_i)$ converges to a set containing all feasible, non-dominated simple paths from v_i to v_c. Finally, when no edges are left to be relaxed, we can choose in Algorithm 1, line 14 from all candidate paths in $R(v_s)$ the one with the minimal cost L, which constitutes the optimal solution.



Linearized ODE model:

$$\underbrace{\begin{bmatrix}\dot{\varphi}(t)\\ \ddot{\varphi}(t)\end{bmatrix}}_{\dot{x}(t)} = \underbrace{\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}}_{A} \underbrace{\begin{bmatrix}\varphi(t)\\ \dot{\varphi}(t)\end{bmatrix}}_{x(t)} + \underbrace{\begin{bmatrix}0\\ 1\end{bmatrix}}_{B} u(t)$$

Figure 5.5: Simple inverted pendulum system with linear differential equation model. The state comprises angle φ and angular speed $\dot{\varphi}$. The plant's input u is the torque applied to the pendulum.

5.7 Evaluation

In this section, we evaluate the performance of our approach using network simulation. In detail, we evaluate the effectiveness of our NCT-service for networked control, the efficiency of network optimization using our routing algorithm, and the runtime performance of this routing algorithm. We begin by introducing our evaluation set-up, before we present the results.

5.7.1 Simulation Environment

For the first two studies, we simulated the integrated NCS consisting of the network, the control system, and the previously described network control logic under different scenarios. We used the OMNeT++/INET simulation environment [VH08] for the event-based simulation of the communication network. The controller synthesis and stepwise simulation of the control system were implemented using the GNU Octave scientific computing language [EBH08], which was integrated with the network simulator using Octave's C++ API and runtime library.

5.7.1.1 Simulation of the Control System

For our evaluation of the closed-loop system, we simulated a textbook example of an unstable physical system, namely, the inverted pendulum system depicted in Figure 5.5. The state of that plant is given by the angle φ of the pendulum with respect to the upright position, and its angular speed $\dot{\varphi}$. The pendulum is in equilibrium in the upright position $\varphi = \dot{\varphi} = 0$, but will tip over if disturbed; it is therefore an unstable system. In the context of our investigated NCS set-up presented in Figure 5.1, one might assume, e.g., that we measure the angle and

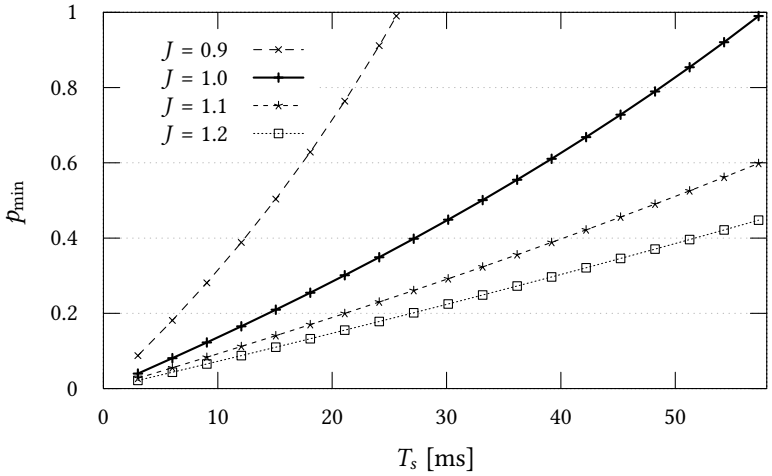


Figure 5.6: Numerically obtained minimum arrival probability function $p_{\min}(T_s)$ for the inverted pendulum shown in Figure 5.5 at different performance levels of J.

speed of the pendulum using a remote camera, which sends its state measurement to a controller connected to the actuator at the base of the pendulum. Our goal is to efficiently apply a torque u depending on the state such that it is kept in an upright position despite disturbances, which are modelled as additive Gaussian white noise.

The controller design method from [BA13] allows us to predict the expected control cost J for this system depending on the parameters p_a and T_s. For the purpose of our simulation study, we prescribe a performance bound of $J_{\max} = 1$. From this we derive the minimum arrival probability function, which for this system is shown in Figure 5.6, as explained earlier in Section 5.5. The figure shows that the sampling period should be no longer than approximately 57 ms if the quality requirement is to be satisfied. We also added plots of $p_{\min}$ for other values of $J_{\max}$ for comparison.

5.7.1.2 Simulation of the Network

In order to simulate our NCS, we implemented two applications for OMNeT++: a `PlantApp` containing the sensor, and a `ControllerApp` implementing the controller as described in Section 5.3.1. Both applications are executed on different hosts in the IP network. The network configuration is performed by a separate

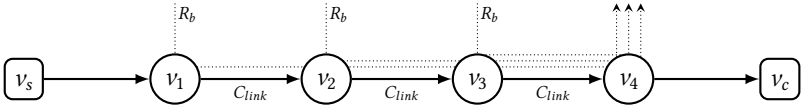


Figure 5.7: Linear topology used to study the validity of our network abstraction

module (`Config`) which maintains a topology model, as well as loss probabilities and latency distributions in the form of discrete empirical CDFs with 100 bins each. These distributions are calculated based on queue time statistics from the network interfaces of the switches. To initially sample these distributions, each experiment is preceded by a 5 s monitoring period during which the `Config` module measures delay and loss. It then executes the optimal routing algorithm, installs the appropriate routing table entries for the NC-flow, and initializes the NCS simulation with the appropriate sampling time T_s.

5.7.2 Effectiveness of NCT Service

In order to evaluate the effectiveness of our NCT-service, we simulated the inverted pendulum NCS introduced above with a 5-hop linear topology shown in Figure 5.7. Nodes v_s and v_c host the `PlantApp` and `ControllerApp`, respectively, and nodes $v_1 - v_4$ are intermediate switches. At each switch $v_1 - v_3$, we inject random UDP traffic with uniformly distributed inter-arrival times and mean bit rate R_b to introduce varying queueing delays and packet losses due to tail drop queueing. As the routing algorithm cannot switch to a lightly loaded path in this topology, we can observe the effective application performance under different levels of network congestion. To this end, we simulated the system for different values of R_b, and measured the average control performance J and load L induced by the NCS traffic.

Figure 5.8 shows these quantities in dependence on the cross-traffic bit rate as a fraction of the link capacity C_{link}. As we can see, the use of the sampling rate determined by our QoS model is effective at satisfying the targeted quality constraint $J_{\max} = 1$ for a large range of cross-traffic levels up to $R_b \approx 0.6 \cdot C_{\text{link}}$. (Note that already a cross-traffic bit rate of $R_b \geq 0.5 \cdot C_{\text{link}}$ leads to full average utilization of the links from v_2 to v_4.) Moreover, performance degrades gracefully under increasing congestion beyond this value. In the case of the inverted pendulum this means that the maximum angle and/or actuation energy increase, cf. (5.5). However, as J remains bounded, the stability of the pendulum is still guaranteed. We can also observe how the NCS traffic load L rises gradually due to the adaptation of T_s.

As the inference of end-to-end latencies is based on the assumption that the

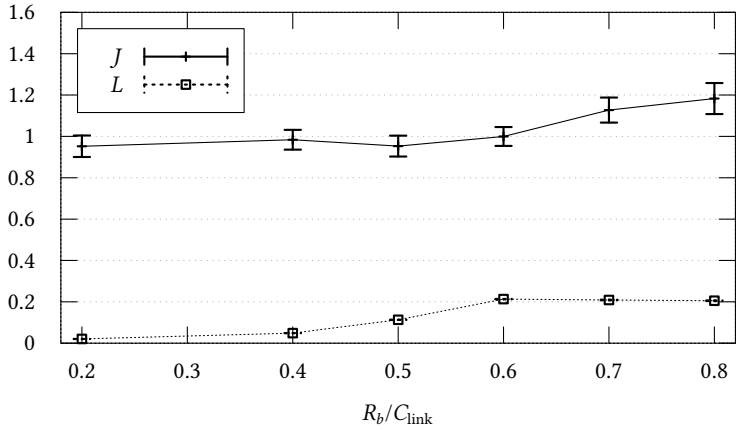


Figure 5.8: Sample mean of measured control cost J and load L of the NCS traffic depending on the data rate of *regular* (i.e., approx. i.i.d.) cross-traffic sources with 95% confidence interval.

link latencies are i.i.d., we also investigated the effects of this assumption being violated, i.e., with strongly correlated traffic. We repeated the above experiment with a mix of regular and bursty (on-off) cross-traffic in equal parts, with traffic bursts lasting 1 s on average. Figure 5.9 shows that, also for bursty traffic, we could reach the targeted NCS performance for cross-traffic levels up to $R_b \approx 0.5 \cdot C_{\text{link}}$, i.e., when the segment from v_2 onward is already fully utilized on average. Beyond this value, the performance degradation of the NCS due to congestion becomes more pronounced. This is to be expected given the higher sensitivity to varying losses for lower sampling periods (as can be seen from Figure 5.6).

5.7.3 Effectiveness of NC Routing

In order to investigate the effectiveness of NC routing, we simulated the same NCS with the topology shown in Figure 5.10, which offers two possible routes from v_s to v_c. From the beginning of each experiment, we injected at each switch random UDP traffic with uniformly distributed inter-arrival times inducing approximately 80% link utilization on all links, except for the link from v_1 to v_4, which is fully utilized on average. As a consequence, the NC-flow is initially routed over the shorter and less heavily loaded path $\left(v_s, v_1, v_2, v_3, v_c\right)$.

At $t = 20\,\text{s}$ after the initial connection set-up, an additional traffic source was added from v_2 to v_3, effectively doubling the utilization of this link. As a

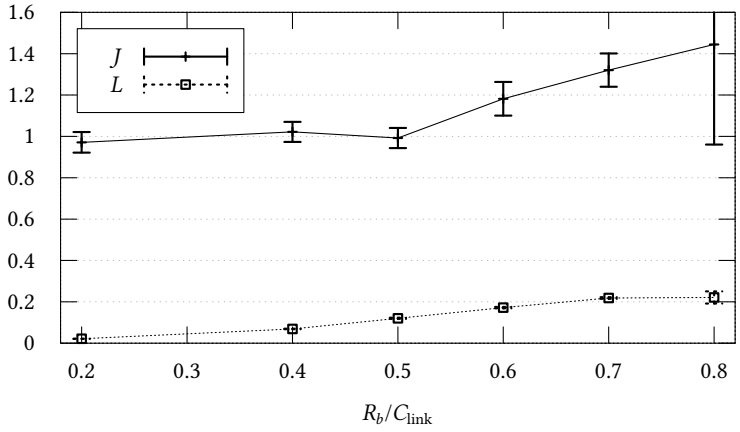


Figure 5.9: Sample mean of measured control cost J and load L of the NCS traffic depending on the data rate of *bursty* (i.e., non-i.i.d.) cross-traffic sources with 95% confidence interval.

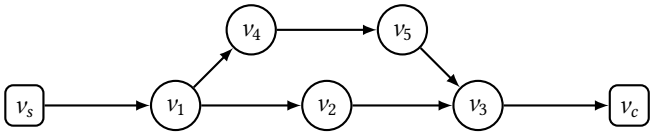


Figure 5.10: Network topology used to study the effectiveness of our routing algorithm

consequence, the arrival probability of this link decreases and the QoS requirement is violated, which is detected by estimating the effective arrival probability of the established NC-connection at the transport layer using an exponential moving average of the inter-loss times (where a loss is equivalent to a `receive()`-timeout). When $p_\sigma(T_s)$ degrades by more than 10%, the system adapts the NC-connection by calculating a new optimal route σ and sampling period T_s. We compared this also with the more naïve strategies of a) only performing transport-layer adaptation of the sampling period $T_s = T_{\max}(\sigma)$ on the previous path σ according to the current effective arrival probability function, i.e., assuming that we must rely on shortest-path routing, and b) adapting neither the route nor the sampling period.

The results of these experiments are shown in Figure 5.11. Here, we show the average application cost J and network load L of the NC-flow for the three approaches described above. As we can see, the application requirement $J_{\max} = 1$

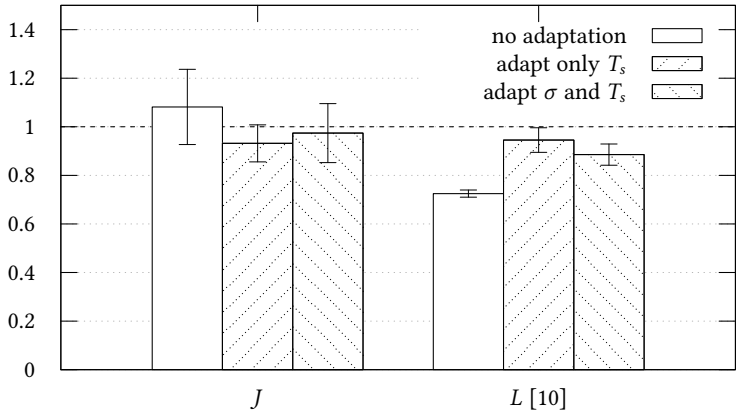


Figure 5.11: Sample mean of measured control cost J and load L of the NCS traffic depending on the use of transport-layer and network-layer adaptation with 95% confidence interval

can be maintained by decreasing T_s (and thus also increasing the load) according to the current network conditions in this experiment. However, by also choosing an optimal route, the load incurred by the NC-flow is reduced by 6%.

5.7.4 Runtime Performance of NC Routing

To show the practicability of our routing algorithm, we evaluated its execution time on a standard PC server with 4×2.7 GHz CPU and 4 GB main memory. We again used the $p_{\min}$ function of the inverted pendulum to specify QoS, and three different network models G using topology and latency measurements of real autonomous systems collected by Rocketfuel [Mah+02]. Specifically, we ran our experiments on AS 1755 (Ebone EU, 87 routers), AS 3257 (Tiscali EU, 161 routers), and AS 1239 (Sprint US, 315 routers)[3]. As the dataset only contains propagation delays with millisecond accuracy, we uniformly randomized the propagation delays at the sub-millisecond range in each experiment. We also added synthetic randomized queueing delays following a Weibull distribution with shape parameter $k \in [0.6, 0.82]$ and mean delay between 1.4 μs and 25 μs (both drawn uniformly). These values are based on the analysis of single-hop empirical delay distributions measured on an IP backbone network in [Pap+03]. The location of v_c was randomly chosen from the set of leaf nodes in each experiment.

[3] AS 1239 is the largest network in the Rocketfuel dataset [Mah+02].

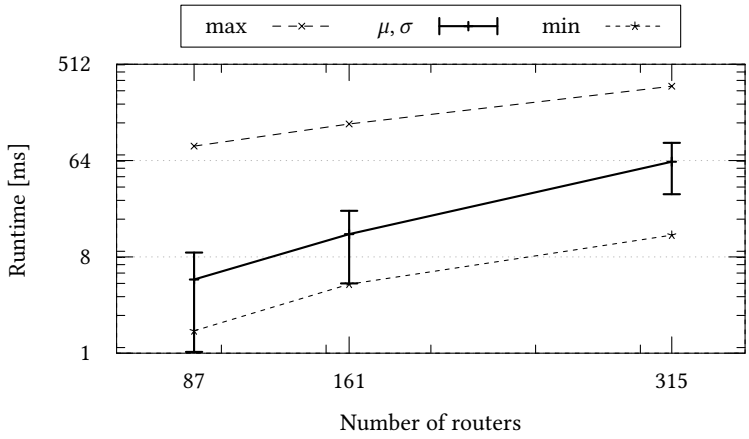


Figure 5.12: Runtime evaluation showing maximum, minimum, and mean (μ) runtime and standard deviation (σ) for 100 000 realizations on three different AS topologies.

We repeated the experiment 100 000 times on each topology. The resulting runtime statistics are shown in Figure 5.12, plotted against the number of routers in the underlying topologies. The 95% confidence intervals for all sample means are narrower than 0.4 ms, and are therefore not visible in the figure. The results indicate that the average and worst-case execution times grow exponentially with the number of nodes in realistic topologies. Nevertheless, with an average of 62 ms and maximum of 319 ms for finding the exact solution to the routing problem on the largest AS topology, our algorithm is certainly practicable for the targeted network size of one AS. For instance, taking our inverted pendulum example and assuming a fairly high sampling frequency of 100 Hz (T_s = 10 ms), we can calculate a new optimal route within 32 samples in the worst case. Note that, due to the required arrival probability of approximately 14% at this sampling period (cf. Figure 5.6), no more than an average of five measurements even have to arrive within this period of time to maintain the QoS requirement.

Moreover, scaling the number of NCS in the network is easily possible since our algorithm lends itself to horizontal distribution to several machines. In this case, the only data structure shared by parallel NCS instances is the global network graph, which is updated only by network monitoring. Thus, the network graph can be easily replicated for each NCS instance.

One critical remaining situation are network failures, e.g., link failures. Typically, the time from detecting a failure to installing a new route might be in the

order of seconds due to monitoring, processing, and route installation delays, which is too slow for many NCS. Here, additional concepts like redundant paths and also replicated controllers (e.g., in different data centers) should be explored.

5.8 Discussion

Finally, we would like to discuss the properties of our approach and possible directions for extensions.

Incremental Deployment As the slow adoption of highly anticipated networking technologies such as IPv6 shows, it is important to discuss the possibility to deploy our approach in practice. As our network architecture in Section 5.4 shows, the NC-service requires adaptions of the transport protocol implementation at the hosts, which is not problematic since hosts are managed by the NCS user. Our transport layer service could even be implemented over UDP on the application layer using a library or middleware, similar to RTP.

On the network layer, we utilize SDN to directly manipulate forwarding tables, which is readily supported by the OpenFlow standard. We think this is a good example how the separation of control plane and data plane enables new protocols and facilitates the implementation of custom, application-aware control logic.

Monitoring Our approach requires information about latencies and loss probabilities, which are dynamic in a best-effort network due to dynamic queue lengths in switches, and congestion due to (changing) cross traffic. While monitoring the network is not a trivial task, a number of state-of-the-art technologies such as PTP as well as existing (active and passive) monitoring systems [RDG14; ADK14; Yu+13] can be applied to this end. However, the overhead and parameters of monitoring (e.g., sampling frequency of latency measurements) should be taken into account.

TCP Friendliness The NC-service adapts sampling rates to network conditions *according to the requirements* of the NCS. In general, it tries to keep the sampling rate and implied network load minimal such that the application requirements are met. However, it does *not* consider the requirements of other flows, in particular, TCP connections competing for bandwidth along shared links. Therefore, NC-connections might get less or more than their fair bandwidth share. TCP-friendliness has not been a design goal of our approach so far. One might argue that real-time critical NC-flows should have priority over elastic TCP flows. However, our evaluation shows that route selection is effective in reducing network load and thus countering possible detrimental effects of sampling rate adaptation.

Nonetheless, congestion control mechanisms limiting the maximum sampling rate could be incorporated.

5.9 Summary

In this chapter, we presented a novel approach for networked control systems where sensors and controllers are communicating over best-effort IP networks with varying latency and packet loss characteristics. We proposed a communication service for NCS, called NC-service, that provides an end-to-end transport abstraction based on a novel probabilistic quality of service specification. On the one hand, this specification is tailored to the requirements of the NCS, giving control system engineers the possibility to readily incorporate it into their familiar analytic frameworks. On the other hand, it links control system performance to network relevant metrics (packet loss and delay). In more detail, this QoS specification provides the network with application-specific knowledge about the relation between application performance and packet loss and delay, and enables the network to optimize routes of NCS flows such that the required application performance is met with minimum network bandwidth resources. By reducing this to a routing problem, we could apply state of the art software-defined networking concepts to implement our custom network control logic based on a logically centralized SDN network controller. Our proposed network architecture lends itself to an easy deployment in SDN-based networks, and is a good example of how SDN eases the integration of network and application.

6 Replication

6.1 Introduction

While the previous chapters have been focused on the communication system and its reliability and performance in terms of QoC, the reliability of an NCS depends on all of its components, i.e., not only the network, but also sensors, actuators, and the controller. The fault-tolerance of a control loop with regard to sensor, actuator, and network failures can be addressed at the application layer, by employing a control design that is inherently robust to a certain degree of sensor and actuator message loss [HNX07; Sch+07; ZJ08]. Such design approaches typically yield some (probabilistic) bounds for the necessary availability of the involved components that ensures stability or a certain degree of performance.

In order to meet these availability requirements, the NCS may have to be designed with redundant components. The availability of sensors can be increased through fusion of redundant sensors [KPS02; LG04]. Similarly, the availability of actuators can be increased by suitable redundant designs, e.g., [Ben+12]. In this chapter, we focus on the availability of the *controller* function.

In order to avoid a single point of failure, the controller has to be replicated over different nodes. In each control step, the controller replicas receive the sensor input and apply the control function to generate the controller output, which is then transmitted to the actuators. Obviously, an important requirement for a replication scheme is to ensure *output consistency* [Ché+92], meaning that all controller replicas send the same sequence of output messages. In this chapter, we use a relaxed output consistency condition like in [Saa+17], where we allow replicas to occasionally produce no output. This is motivated by the fact that intermittent message loss in NCS—also from controller to actuator—is well-studied [Sch09; ZY10; QN12].

In general, output consistency could also be achieved by following the state machine replication (SMR) approach for implementing the controller function, where each control step corresponds to calling an operation that takes the sensor values as input and generates the controller output. In this case, a standard SMR protocol such as Paxos [Lam98], Viewstamped Replication [OL88], or RAFT [OO14] could be used to achieve consensus about the controller outputs. Unfortunately, as pointed out in [Saa+17], SMR is unsuitable for controller replication due

to the time-constraints of NCS. Since each control step has to be finished within a certain time bound, the replicas only have limited time to achieve consensus on each output. If consensus cannot be achieved in time, no output can be sent. While a fault-tolerant controller can accept missing outputs in some control steps, the problem with SMR is that a replica cannot start processing a new control step before having completed the previous one. Therefore, a delay violation of consensus processing in one control step affects all the following steps.

In this chapter, we investigate how the problems of SMR subject to real-time constraints can be avoided when consensus algorithms are used for controller replication. To this end, we first discuss the necessary consistency concepts for replicated control systems. While output consistency addresses the *spatial* correctness concerning the set of messages delivered to the actuators in one time step, it does not relate to any notion of *temporal* correctness regarding the *sequence* of controller outputs. However, any stateful control law specifies such temporal behaviour. Since the controller's dynamics is part of the closed-loop model, which is the basis for stability and performance analysis, it should be reproduced faithfully by the replicated controller. In other words, output consistency alone is not sufficient to ensure *replication transparency* for NCS.

Therefore, we argue that it should be complemented by an additional condition, which we call *state consistency*, that requires any new output to be based on the controller state that generated the most recent previous output. State consistency ensures that the replicated controller produces a sequence of outputs that appears to have been generated by a single controller that always executes uninterruptedly, and that any intermittent unavailability of the controller is indistinguishable from an omission of output messages. Consequently, the properties that have been proven for a non-replicated controller also hold for a replicated version of the same controller, with no need for modifying the controller design, assuming that the non-replicated controller is designed to tolerate output omissions to a degree that is equivalent to that achieved by the replicated controller for a given network. While SMR offers equivalent properties in the failure-free case, state consistency provides a well-defined correctness condition also for *incomplete* controller output sequences, which facilitates the temporal decoupling of consensus processing for consecutive control steps to avoid the problems of SMR.

We propose the State-Consistent Replication Management (SCRAM) protocol for NCS controllers. For each iteration of the replicated control function, a single-value consensus algorithm based on a standard protocol [CT96; Dol+96] is executed to ensure output consistency. However, SCRAM ensures that the replicated controller is not blocked unnecessarily waiting for consensus instances to terminate, and that the state of controller replicas producing new output is based on the state of replicas that have produced output previously, thus ensuring state consistency. Moreover, we show that SCRAM is competitive with—and in

many cases outperforms—existing real-time replication protocols that only ensure output consistency in terms of availability. While generic performance metrics such as latency and availability of the replicated controller influence QoC to a major extent, the application-specific quality of the data produced by the replicated controller is also an important factor in determining QoC, especially when comparing different replication protocols with similar availability characteristics.

In summary, this chapter contains the following contributions:

- A formal definition of state consistency for replicated controllers, providing sufficient guarantees to make a replicated controller functionally indistinguishable from a non-replicated controller.
- SCRaM, a replication management protocol for NCS ensuring state and output consistency with high availability.
- An experimental comparison of SCRaM with an existing protocol that demonstrates the benefits of state consistency.
- A case study for replicated linear quadratic Gaussian (LQG) controllers for LTI systems using SCRaM.
- An application-specific metric for evaluating different controller replica executions with respect to the closed-loop QoC.
- Application-specific modifications of SCRaM based on this metric, for increasing QoC.

The remainder of this chapter is organized as follows. In Section 6.2 we discuss related work. In Section 6.3 we describe a generic model for time-triggered control systems supporting replication. In Section 6.4 we formalize the replication transparency problem for control systems. In Section 6.5 we extend the system model to describe a group of replicated controllers and derive the output and state consistency conditions. In Section 6.6 we derive requirements for a replication protocol to satisfy these consistency conditions and present the SCRaM protocol. Evaluation results are presented in Section 6.8. Section 6.9 concludes the chapter with a short summary and outlook.

6.2 Related Work

It is well known that in *synchronous systems* with bounded network and processing delay, consensus can be reached in bounded time, even for stronger failure models than we assume in this chapter, such as Byzantine failures [LSP82]. Therefore, some replicated systems rely on real-time communication networks such as

Table 6.1: Notation

Indexing is denoted by bracket notation, so $x[i]$ indicates the i^{th} component of a vector x. In most cases, subscripts denote sampling periods, i.e., time, and superscripts denote replicas, so x_k^r indicates the value of (a vector) x on (or associated with) replica r at time t_k. Undefined values are denoted by $\perp$. The set $\mathbb{R}_\perp = \mathbb{R} \cup \{\perp\}$ is used to denote possibly missing scalars. In pseudocode, an ampersand preceding arguments indicate call-by-reference semantics, i.e., `&var` denotes a reference to the variable `var`.

Time-Triggered Ethernet in the case of the DFT4FTT project [Bal+18] or Token Ring in the case of the Delta-4 project [Ché+92], adding sufficient redundancy in the network to practically avoid delay failures, which would otherwise violate the fundamental assumption of a synchronous system. The middleware Achal [GAB19] is targeted specifically at CPS by providing a distributed timestamped key–value store based on a Byzantine fault tolerance (BFT) algorithm for synchronous networks. However, real-time networks with deterministically bounded delay are mostly restricted to local or possibly metropolitan area. Therefore, we cannot assume a synchronous system to be available for CPS distributed over a larger geographic area, where one has to communicate over unreliable Internet connections.

Considering that in asynchronous systems one cannot guarantee both, agreement and termination within bounded delay, Saab et al. recently proposed the Quarts protocol [Saa+17]. It guarantees that in each time step, replicated controllers agree on a single output vector sent to actuators (output consistency) through plurality voting. The delay of attempting agreement is bounded such that non-agreement in one step cannot indefinitely block the next steps. The authors show that Quarts is superior to the classic asynchronous consensus protocol FastPaxos in terms of availability and latency. However, Quarts does not address the issue that the agreed actuator values of one step should also be consistent with the agreed values of another round. In contrast, in addition to output consistency, we enforce state consistency to ensure that the state of a replica producing the agreed output values of the current round is based on the state of the replica that has produced the previously agreed values. This makes the functional behaviour of the replicated controller indistinguishable from a non-replicated controller. This fundamentally simplifies the design and analysis of the replicated controller based on a functionally equivalent non-replicated controller.

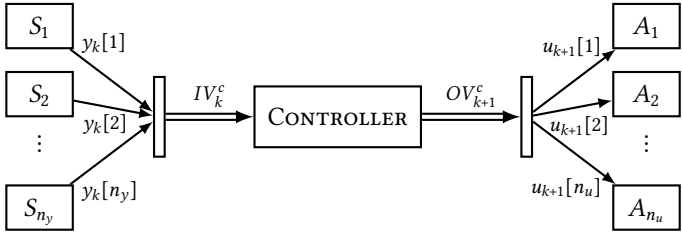


Figure 6.1: Set-up showing sensors, controller, actuators and communication channels. The physically connected plant is omitted for clarity.

6.3 Control System Model

The basic controller set-up is shown in Figure 6.1. The controller receives the measurements from multiple sensors (denoted as its *input vector IV*), which it uses to update its internal state and calculate values for all actuators (comprising its *output vector OV*). The controller communicates with the sensor and actuator components over a packet-switched network. Therefore, the input and output signals of the controller are discrete-time sequences of input and output vectors.

Accordingly, we assume that the plant is described by some dynamical system model that is sampled periodically at times t_k, which are determined by a sampling interval $T_s = t_{k+1} - t_k$. We denote the measurements from all sensors at sampling time t_k by y_k, and the actuator values by u_k.

The controller maintains its own internal state S_k^c. As described above, the controller periodically updates its state depending on the values received by the sensors, and generates a corresponding vector of actuator values. Because the sensor values are transmitted over a communication network, values may be lost or arrive too late, i.e., the input vector $IV_k^c \in \mathbb{R}_\perp^{n_y}$ received by the controller may contain only a subset of the sensor values $y_k \in \mathbb{R}^{n_y}$, such that $IV_k^c[i] = \perp$ if $y_k[i]$ was not received. The input vector is used to update the controller's state $S_k^c \longrightarrow S_{k+1}^c$, from which an output vector $OV_{k+1}^c \in \mathbb{R}^{n_u}$ is then generated and sent to the actuator nodes. The actuator vector $u_{k+1} \in \mathbb{R}_\perp^{n_u}$ may contain only a subset of the output vector components, again, due to delay and loss. More specifically, each actuator node is responsible (w.l.o.g.) for one component $u[i]$ of the actuator vector. Due to the real-time behaviour of the control system, the actuator node applies $u_{k+1}[i] = OV_{k+1}^c[i]$ if it receives the output vector by time t_{k+1}, otherwise the corresponding component $u_{k+1}[i] = \perp$ is undefined. The execution model of the controller is outlined in Figure 6.2.

Note that the methods for dealing with missing values in control systems depend on the specific control design and are therefore beyond the scope of this

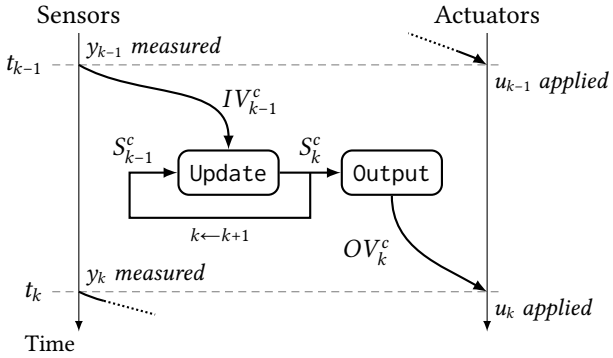


Figure 6.2: Controller execution model for sampling period k

thesis. We refer to the literature for a discussion of missing sensor [LG04] and actuator [Sch09] values in NCS.

The state and output of the controller evolve according to

$$S^c_{k+1} = \texttt{Update}\left(S^c_k, IV^c_k\right), \tag{6.1}$$

$$OV^c_{k+1} = \texttt{Output}\left(S^c_{k+1}\right). \tag{6.2}$$

We assume that both functions are deterministic. For convenience of notation, we use the following shorthand for the repeated `Update` function:

$$\begin{aligned} &\texttt{Update}^{n+1}(S, IV_0, IV_1, \ldots, IV_n) \\ &= \texttt{Update}^{n}\left(\texttt{Update}(S, IV_0), IV_1, \ldots, IV_n\right), \end{aligned} \tag{6.3}$$

where $\texttt{Update}^0(S) = S$. In the following, unless noted otherwise, we consider *period* k to be synonymous with the time interval $(t_{k-1}, t_k]$, i.e., the time interval during which the controller updates its state to S_k.

6.4 Problem Statement

Clearly, the behaviour of the NCS depends on the sequence of actuator values u_k delivered to the actuators. As mentioned above, an output vector sent by a controller may be lost due to faulty communication channels. Consequently, even in the case of a perfectly reliable controller, communication failures may leave actuators with undefined values. Since the possibility of missing actuator values is inevitable, we introduce the following definition.

Definition 6.1 (Faithful Controller)**.** A controller is *faithful* if it maintains a state that evolves according to (6.1) but may omit output vectors sporadically, i.e.,

$$OV_k^c \in \left\{ \texttt{Output}\left(S_k^c\right), \perp \right\}. \tag{6.4}$$

The notion of a faithful controller facilitates the definition of state consistency in Section 6.5 by allowing us to account also for a controller (replica) being unable to send an output (reach agreement) due to node failures or deadline misses. Most importantly, subject to faulty communication channels, the *values* of output vectors that are received by the actuators from a faithful controller are indistinguishable from those of a perfectly reliable controller. In short, faithful behaviour of the replicated controller ensures *replication transparency* [CDK01].

It is important to note that key properties of control systems—such as stability, performance, and robustness w.r.t. disturbances—are sensitive to the end-to-end loss and latency characteristics of the entire control loop, which should always be taken into account, also when designing an NCS with replicated controller. For instance, if the the non-replicated controller is designed under the assumption of i.i.d. losses (including deadline violations) with a certain distribution, then the closed-loop system with the replicated controller can only be guaranteed to behave equivalently if it also exhibits i.i.d. losses with the same distribution. This is sometimes referred to as *performance transparency* [Cro96]. However, these characteristics are not determined by the (replicated) controller alone but by all CPS components, i.e., sensors, actuators, and the network.

In this chapter, we focus on the replicated controller, making as few assumptions as possible on the other components. From this perspective, we stress that faithful behaviour of the replicated controller is a *necessary* condition for the replication-agnostic NCS design, independent of the control design method being used. While modelling end-to-end losses is an orthogonal but equally important design prerequisite, it is beyond the scope of this chapter. Its treatment for special cases (i.e., particular control system, component failure, and network models) should be considered in future research.

6.5 Consistency Models

Now, we consider a group of N replicated controllers, which we denote by the set G. All controllers execute the same control functions, i.e., `Update` and `Output`. However, each controller replica $r \in G$ may experience different input vector sequences due to different (partial) arrival patterns of measurement components. Therefore, also the controller states S_k^r and outputs OV_k^r may differ in general among replicas. Moreover, replicas may be unavailable, e.g., due to crash failures,

in which case they do not generate an output. For each period, we denote the subset of *influential* replicas that generate an output OV_k^r in period k as G_k^{out}, i.e.,

$$r \in G_k^{\text{out}} \iff OV_k^r \neq \bot. \tag{6.5}$$

Our goal is to design a replication scheme where the behaviour of a group of replicated controllers is equivalent to that of a faithful controller. In the following, we describe two consistency models which together provide this equivalence.

Definition 6.2 (Output Consistency)**.** A controller group G is called *output consistent* if

$$\forall_{k>0} \forall_{r,s \in G_k^{\text{out}}} \quad OV_k^r = OV_k^s, \tag{6.6}$$

i.e., all influential replicas generate the same output.

Output consistency guarantees that the group does not send "conflicting" commands to different actuators at any time (nor that any actuator receives several different commands). To illustrate the importance of output consistency in the context of NCS, let us consider tracking control for a milling machine as a simple example. Each controller estimates the current position and speed of the milling head and determines as its output a vector of forces $[F_x, F_y]$ along the x and y axes required to keep the tool moving along the desired trajectory. Assume that the target trajectory is a line at 45° and two controller replicas estimate the same position but slightly different speeds, leading to two different control outputs that both represent a force tangent to the trajectory, say $[0.9, 0.9]$ and $[1.1, 1.1]$, respectively. If the corresponding actuators should now receive outputs from *different* replicas, e.g., $F_x = 0.9$ and $F_y = 1.1$, the resulting force (at approximately 50.7°) is clearly not in alignment with the desired trajectory, at least in the current time step, leading to a reduced quality.

Note, however, that Definition 6.2 does not make any statements about the sequence OV_k itself *over time.* Since a faithful controller updates its state according to (6.1) in every period, any output vector is based on a state that can be obtained through a series of `Update` steps from the state on which the previous output vector was based, even if an arbitrary number of output vectors were omitted in the interim. We say that the corresponding state is *reachable* from the state from which the previous output was generated according to the following definition.[1]

Definition 6.3 (Reachability)**.** The set of states S_{k+n} in period $k+n$ that are *reachable* from a state S_k in period k is $\mathcal{R}_n(S_k) = \left\{ \texttt{Update}^n(S_k, IV_k, \dots, IV_{k+n-1}) \;\middle|\; IV_\tau \in \mathcal{L}(y_\tau) \right\}$.

[1] This definition differs from the usual reachability notion in control theory in that the input sequences are constrained.

Here, $\mathcal{L}(y)$ is the set of possible input vectors received for the measurement vector $y \in \mathbb{R}_{\perp}^{n_y}$ transmitted over the network, i.e., $IV \in \mathcal{L}(y) \iff \forall_i IV[i] = y[i] \vee IV[i] = \perp$. We are now ready to introduce the stronger concept of state consistency, which reflects the state interdependencies of a controller group that behaves faithfully.

Definition 6.4 (State Consistency)**.** A controller group G is called *state consistent* if

$$\forall_{k\geq 0}\forall_{r,s\in G_k^{\text{out}}} \quad S_k^r = S_k^s \tag{6.7}$$

$$\forall_{k>0}\forall_{r\in G_k^{\text{out}}}\forall_{s\in G_{k-d(k)}^{\text{out}}} \quad S_k^r \in \mathcal{R}_{d(k)}\left(S_{k-d(k)}^s\right) \tag{6.8}$$

where $d(k) = \min\left\{ \tau \;\middle|\; \tau > 0,\ G_{k-\tau}^{\text{out}} \neq \emptyset \right\}$, i.e., all influential replicas share a controller state that is *reachable* from the controller state of the *most recent* influential replicas.

State consistency implies output consistency, since the state S_k^r determines the output OV_k^r, cf. (6.2). To illustrate the importance of state consistency, let us again consider the milling example from earlier. One important concern in machining (or other mechanical systems such as aircraft [Lev+00]) is the suppression of so-called chatter, which may impair quality or even lead to actuator damage, and controllers are often designed accordingly [Doh+04]. In our example, we therefore assume that each controller replica produces an output sequence where the rate of change in actuator force is limited. However, without state consistency the replicated controller could alternate in the worst case between outputs from both replicas such that the magnitude of the output oscillates, leading to an undesirable chattering motion of the actuator. In general, violation of state consistency may introduce (high-frequency) dynamics that amount to an *unmodelled* disturbance that was not considered during the controller design.

By contrast, the output sequence produced by a controller group that satisfies Definition 6.4 is equivalent to that of a faithful controller. Since a faithful controller can omit to send output vectors, we can show equivalence by considering output vectors in periods where $G_k^{\text{out}} \neq \emptyset$.

Theorem 6.1. *Consider a controller group G with uniform initial controller state $\forall_{r\in G}\, S_0^r = S_0$ and a faithful controller c with initial controller state $S_0^c = S_0$. If G is* state consistent*, then there exists a sequence of input vectors $IV_k^c \in \mathcal{L}(y_k)$ for the faithful controller such that*

$$\forall_{k>0}\forall_{r\in G_k^{out}} \quad OV_k^r = OV_k^c, \tag{6.9}$$

i.e., the outputs OV_k generated by the group G are identical to the corresponding outputs generated by a faithful controller.

Proof. We use inductive reasoning to show that there exists a sequence $\left(IV_k^c\right)_{k\geq 0}$ of input vectors for which (6.9) is satisfied. Since the output of c obeys (6.4), we can assume $G_k^{\text{out}} = \emptyset \implies OV_k^c = \bot$ without loss of generality. Let the periods when G produces an output be denoted by $\left(k_i\right)_{i\geq 0}$, where $k_{i+1} = \min\left\{\tau \mid \tau > k_i,\ G_\tau^{\text{out}} \neq \emptyset\right\}$, $k_0 \triangleq 0$.

Basis Since the initial state S_0 is fixed, the only possible initial output is $\forall_{r\in G_0^{\text{out}}}\, OV_0^r = OV_0^c = \texttt{Output}(S_0)$. Hence, (6.9) is satisfied for $k \leq k_0 = 0$ with an empty input vector sequence $\sigma_0 = \varepsilon$.

Inductive Step Assume that (6.9) is satisfied for $k \leq k_i$ with some input sequence $\sigma_i = \left(IV_k^c\right)_{k=0}^{k_i-1}$ and that all replicas $s \in G_{k_i}^{\text{out}}$ hold the same state $S_{k_i}^s \equiv S_{k_i}^c \triangleq S_{k_i}$ as c.

Now consider a replica $r \in G_{k_{i+1}}^{\text{out}}$ and note that $k_{i+1} = k_i + d\left(k_{i+1}\right)$. Because G is state consistent, $S_{k_{i+1}}^r$ is reachable from S_{k_i}, cf. (6.8). Therefore, there exists a sequence of input vectors $\sigma' = \left(IV_k'\right)_{k=k_i}^{k_{i+1}-1}$ such that

$$S_{k_{i+1}}^c = S_{k_{i+1}}^r = \texttt{Update}^{d(k_{i+1})}\left(S_{k_i}, \sigma'\right)$$

and, consequently, $OV_{k_{i+1}}^c = OV_{k_{i+1}}^r$. Because of (6.7), this is true with identical values for all replicas $r \in G_{k_{i+1}}^{\text{out}}$. Therefore, (6.9) is satisfied for $k \leq k_{i+1}$ with the input sequence $\sigma_{i+1} = \sigma_i \circ \sigma'$. ■

6.6 Replication Algorithm

Our goal is to design a replication protocol for a group G of replicated controllers whose behaviour is equivalent to that of a faithful controller. As we have shown, a sufficient condition for this is that the underlying replication protocol offers both output and state consistency. Before describing the SCRAM protocol in detail, we first present an outline of the algorithm and discuss the requirements it should satisfy.

6.6.1 Outline and Requirements

First and foremost, replicas must implement the execution model of the generic controller as described in Section 6.3. Hence, the basic algorithm executed by each replica consists of a *controller loop* that repeats the following steps:

1. receive an input vector IV_k,

2. update the controller state,
3. calculate the output vector and send it to the actuators,
4. wait for the next sampling time t_{k+1} and go to step 1.

However, since output consistency is required, we need to ensure that all replicas agree on a unique output vector in step 3 of each period. For this reason, we also execute one instance of a *single-value consensus* [FLP85; CT96] algorithm in each iteration of the controller loop, which is the basic building block of our replication scheme.

Let us denote the consensus instance for period k (i.e., for output vector OV_k) as C_k. We consider C_k to be the problem of agreeing on the state S_k, from which the unique output vector OV_k can be calculated deterministically. Any algorithm for solving the single-value consensus problem must satisfy the following standard properties [FLP85; CT96]:

Agreement All correct replicas decide the same value for C_k.

Integrity If a correct replica decides the value S_k for C_k, S_k must have been proposed for C_k by some replica.

Termination Every correct replica eventually decides a value for C_k.

The *agreement* and *integrity* properties ensure output consistency, provided that replicas may only send an output vector after the corresponding consensus instance has been decided.

However, the *termination* property only requires that a decision is made eventually, and is therefore not sufficient for determining a unique output within one sampling period. Indeed, it is impossible to guarantee upper bounds on the time required to complete consensus in asynchronous distributed systems [FLP85] or in synchronous systems subject to omission failures [SWK09]. Because any component of an output vector OV_k produced from a state S_k is discarded if it does not arrive at the corresponding actuator by time t_k, a consensus instance that terminates any later will not provide any useful output. But since output vectors may be omitted by a faithful controller, consensus instances can be aborted at the end of their corresponding sampling period without jeopardizing output consistency.

Conversely, state consistency requires that the controller states from which consecutive outputs are generated satisfy the reachability condition expressed in (6.8). Therefore, whenever a consensus instance *does* terminate within its period, the decided state S_k becomes influential in the sense of Definition 6.4 and all future decided states must be reachable from S_k. We denote such an instance as successful.

Definition 6.5 (Successful Consensus Instance)**.** Let the unique value decided by consensus instance C_k be denoted by $\texttt{Decision}(C_k)$, and the time at which C_k yields its decision be defined as the *earliest* time at which any correct replica decides this value. A consensus instance C_k is *successful* if and only if it yields $\texttt{Decision}(C_k)$ at some time $t \leq t_k$.

While a replica that decides a value S_k can infer that the consensus instance C_k was successful, the opposite is not true, i.e., a replica not deciding by time t_k cannot be sure whether C_k was successful, nor whether its controller state S_k is reachable from the most recently decided state. Since proposing a state that does *not* satisfy this reachability condition could threaten to violate state consistency, we must suitably restrict the set of controller states that may be proposed for subsequent consensus instances.

Based on these observations, we define the following requirements that the controller loop should satisfy with respect to the consensus instance executed in each period.

Requirement 1 (Output Constraint) The output vectors of each replica $r \in G$ must satisfy

$$OV_k^r \in \left\{ \texttt{Output}(\texttt{Decision}(C_k)), \bot \right\}, \tag{6.10}$$

$$OV_k^r \neq \bot \implies C_k \textit{ successful.} \tag{6.11}$$

Requirement 2 (Proposal Constraint) Any replica $r \in G$ may only propose values S_k^r for C_k that satisfy

$$S_k^r \in \mathcal{R}_{k-k'}\left(\texttt{Decision}(C_{k'})\right), \tag{6.12}$$

where $k' = \max\left\{ \tau \;\middle|\; \tau < k, C_\tau \text{ successful} \right\}$. (Assume that there exists a successful consensus instance C_0 with $\texttt{Decision}(C_0) \triangleq S_0$ for the initial state.)

Clearly, Requirement 1 together with the agreement property ensures output consistency since it guarantees that no conflicting output vectors may be generated by different replicas for the same sampling period. Requirement 2 together with the integrity property ensures state consistency since it guarantees that only controller states that are reachable from previous influential controller states can be proposed and hence decided.

Finally, it is worth noting that replicas should seek to start a new consensus instance C_{k+1} as soon as new input from the sensors is available since this maximizes the likelihood of the new instance being successful and producing an output within the corresponding sampling period. This requires replicas to make

a proposal at that point. Requirement 2 already ensures that any such proposal must not contradict the *possible* outcome of the previous consensus instance C_k (and, in fact, any earlier instance). Therefore, each replica can abort the previous consensus instance at the beginning of the new sampling period.

6.6.2 Distributed System Model

For describing the algorithm, we consider an asynchronous distributed system where replicas may suffer crash failures and the network may suffer omission failures. For simplicity, we assume that processing latency is negligible.

Moreover, we assume that every replica is equipped with an unreliable failure detector [CT96], and let $\texttt{Suspects}_r$ denote the set of replicas that r currently suspects to have failed. We assume that the failure detector satisfies strong completeness and eventual weak accuracy [Dol+96], which facilitates the use of an existing consensus algorithm as part of our protocol.[2]

Finally, we assume that clocks are synchronized among sensors, actuators, and controller replicas. Synchronization of sensors and actuators is an implicit assumption of the control system model, while sufficiently accurate synchronization of replicas is useful for obtaining tight bounds on the timeouts on listening for input vectors and sending output vectors.

6.6.3 Algorithm

We now present the SCRAM protocol by explaining the controller loop and the consensus instances executed by each replica in more detail. Algorithm 3 shows the algorithm executed by each replica. Lines 1–7 show the initialization of the necessary variables. As before, k denotes the current sampling period, S the replica's local controller state, and IV its current input vector. In addition, the "base" period k_b indicates the period of the most recent consensus instance C_{k_b} on whose (possible) decision the controller state S is based.[3] This value is updated whenever a replica modifies its state information through consensus and is used to satisfy Requirement 2 as will be shown in the following. The remaining variables, `mode`, v, and v_b, are related to the consensus algorithm and will be explained in the corresponding sections.

At this point we note that, although we defined a consensus instance C_k as a procedure for agreeing on a state S_k for simplicity, our algorithm is designed to

[2] Such failure detectors can be implemented using timeout mechanisms [CT96].

[3] Note that C_{k_b} need not have been successful. Rather, the base period expresses that S is reachable from the most recent influential state $S_{k \leq k_b}$ up to period k_b, while there may have been a more recent influential state $S_{k > k_b}$ from which S is not necessarily reachable.

agree on the tuple (S_{k-1}, IV_{k-1}). However, since the controller state S_k can be calculated deterministically as $\texttt{Update}(S_{k-1}, IV_{k-1})$, both descriptions are equivalent. We advocate the latter form since it leaves open the possibility of exchanging input vectors separately from controller states, either for the purpose of message reduction in the failure-free case when replicas already possess S_{k-1} from the previous period (assuming IV requires less memory than S), or for combining input vectors from multiple replicas in order to fill in missing components like in [Saa+17].

Algorithm 3: SCRAM protocol executed on each replica $r \in G$

```
1  k ← 0;                                                  // current period
2  S ← S_0;                                                // controller state
3  IV ← [⊥, ..., ⊥];                                       // input vector
4  k_b ← 0;                                                // "base" period of estimate
5  mode ← normal;                                          // operation mode
6  v ← 0;                                                  // current view number
7  v_b ← 0;                                                // "base" view (most recent proposal)
   /* ========================= MAIN LOOP ========================= */
8  while true do
9      wait until t ≥ t_k;                                 // await sampling period
10     IV ← ReceiveAndConstructIV(k);
11     k ← k + 1;                                          // advance period counter
12     decided ← false;
13     est ← (S, IV, k_b);                                 // prepare estimate
14     try                                                 // within current period
15         Consensus(k, &est, &mode, &v, &v_b);            // C_k
16         decided ← true;
17     catch timeout at t_k                                // period expired
           // Consensus C_k aborted
18     end
       /* Recover state from estimate and update:                          */
19     S ← Update(est.S, est.IV);
20     IV ← [⊥, ..., ⊥];
21     k_b ← est.k_b;
22     if decided then
23         send OV_k = ⟨Output(est.S), k⟩ to actuators;
24     end
25 end
```

6.6.3.1 Controller Loop

The remainder of Algorithm 3 shows the controller loop executed by each replica. At the beginning of a sampling period, the input vector is received (l. 10), where the function `ReceiveAndConstructIV(`k`)` assembles the input vector IV_k from all received components of the measurement y_k. It returns after a suitably chosen timeout period[4], setting components not received so far to $\perp$. The counter k is incremented (l. 11) after receiving the input vector in order to reflect the active period, i.e., the period of the output vector OV_k to be generated.

At this point, the replica starts the consensus instance. To this end, the quantities that are the subject of consensus, namely S, IV, and k_b are collected into a data structure `est`. This data structure is passed as an argument to the consensus procedure and is called *estimate* in [CT96] since it is a value that may be proposed or decided by the consensus instance, and is therefore the replica's "best guess" of the outcome of the current consensus instance.[5]

In Line 15 the `Consensus` function is called for period k, and a reference to `est` is passed along with references to `mode`, v, and v_b, which are parameters of the consensus protocol described in Section 6.6.3.2. If `Consensus` returns within the current sampling period, the consensus instance was successful and `est` contains the agreed-upon values for S and IV. However, if the current sampling period expires before completion (l. 17), `Consensus` is aborted. While a timeout is used for the sake of presentation, the abort could also be triggered by sensor messages from the next period. (Of course, an abort does *not* imply that the corresponding consensus instance was unsuccessful, merely that no decision was received by the replica.)

Note that, even if `Consensus` is aborted, `est` may have been modified. This is important for ensuring state consistency since aborted instances could have received proposals which were potentially decided. In order to ensure that a decided controller state is used as the basis for the controller state to be proposed (and possibly decided) in the following period, even in the face of crash failures, it must be guaranteed that a majority of replicas are aware of that decided state. While it cannot be guaranteed that a successful decision is received by a majority of replicas within the current sampling period, a decision implies that a majority of replicas received and acknowledged the corresponding proposal, which is then contained in those replicas' estimate at the end of the sampling period.

Therefore, the new controller state S at the end of the sampling period is determined as the `Update` of the state and input vector contained in `est` (l. 19),

[4] based on the network delay

[5] Since we use a leader-based consensus protocol, cf. Section 6.6.3.2, only one replica makes a proposal at a time. However, the estimates of different replicas may be proposed when new leaders are chosen or a different consensus protocol is used.

IV is cleared (l. 20), and the base period k_b of the current state is set to that of the estimate (l. 21). Note that this update is performed regardless of the success of the consensus instance, so that S contains a controller state that is reachable from the decided state of any possibly successful consensus instance C_{k_b}. (If a proposal was received, then $k_b = k$.) Finally, if a decision was received, i.e., `Consensus` was not aborted, then the replica calculates the output vector OV_k and sends it to the actuators (l. 23) before waiting for the next sampling period. Note that multiple replicas might send the *same* output vector to the actuators, thereby increasing the probability that each actuator receives its required value.

6.6.3.2 Consensus

For each instance C_k, we use a modified version of the consensus algorithm by Dolev et al. [Dol+96], which is an adaptation of the algorithm by Chandra and Toueg [CT96, Section 6.2] for considering omission failures. It is shown in Algorithm 4. We strive to keep the presentation close to the original algorithms, with our core modifications—for distinguishing consensus instances of different periods and for achieving state consistency—highlighted in grey. However, we split the algorithm into two parts described separately in the following, in order to better explain the (failure-free) normal operation of the algorithm as opposed to the handling of node failures. Instances of the modified consensus algorithm are aborted after the corresponding sampling period has ended, as described in the previous section.

The protocol uses a dedicated *coordinator* replica that is responsible for deciding on an estimate. In such leader-based protocols, we can distinguish between the *normal* operation mode when there is a single persistent coordinator without failures, and a special *view change* mode where a new coordinator has to take over (e.g., due to a crash failure of the previous coordinator) before normal operation can resume. We will first describe normal operation before discussing the mechanism for changing the coordinator.

Notation In Algorithms 4 and 5, received messages are expected to match a certain format, which is indicated in the pseudocode. Fields marked prime are placeholders for arbitrary values. E.g., the predicate *received* $\langle$`Propose`, k, v, `est`$'\rangle$ specifies that a `Propose` message with the specified values for k and v is expected, while an arbitrary value `est`$'$ for field `est` is accepted. Messages not matching that format are discarded or treated specially where indicated. In particular, messages whose period number does not match the current period k are discarded.

Normal Operation Mode For the moment, we ignore the first lines 3–6, as they do not pertain to normal operation mode. At the beginning of the consensus instance

Algorithm 4: Consensus algorithm executed by $r \in G$ for period k. Modifications w.r.t. [CT96; Dol+96] are shown in grey.

```
1  Function Consensus(k, &est, &mode, &v, &v_b)
2      while true do
3          if mode = viewchange then
4              v ← v + 1;
5              NextView(k, &est, &mode, &v, &v_b);
6          end
7          c ← coord(v);                          // c = current coordinator
8          if r = c then
9              est.k_b ← k;                       // set base period of est
10             send ⟨Propose, k, v, est⟩ to G;
11         end
12         wait until received Propose from c or c ∈ Suspects_r;
13         if received ⟨Propose, k, v, est′⟩ then
14             est ← est′;
15             v_b ← v;
16             send ⟨ACK, k, v⟩ to c;
17         else if received message with v′ > v then
18             v ← v′;
19             NextView(k, &est, &mode, &v, &v_b);
20             continue;
21         end
22         if r = c then
23             wait until received ⌈(N+1)/2⌉ ⟨ACK, k, v⟩;
24             send ⟨Decide, k, v, est⟩ to G;
25         end
26         wait until received Decide from c or c ∈ Suspects_r;
27         if received ⟨Decide, k, v, est′⟩ then
28             est ← est′;
29             v_b ← v;
30             return;                                          // decided
31         else if received message with v′ > v then
32             v ← v′;
33         else                                  // coordinator failure suspected
34             v ← v + 1;
35         end
36         NextView(k, &est, &mode, &v, &v_b);
37     end
38 end
```

C_k, the coordinator c sets the base period $\texttt{est}.k_b$ of its estimate to the period k of the current consensus instance (l. 9) since this estimate is to be proposed and may end up being decided. (This value is used to maintain state consistency when electing a new coordinator, which is described later.) It then multicasts a `Propose` message containing its estimate to G (l. 10). Upon receiving a proposal for the current period k, all replicas (including c) accept it by logging the contained values and reply to c with an acknowledgement (ll. 14–16). After receiving acknowledgements from a majority of replicas, the coordinator confirms the decision by multicasting a `Decide` message (l. 24). Upon receiving a decision for the current period k, all replicas (including c) accept it by logging the contained values (in case the corresponding `Propose` had not yet been received) and return to the controller loop, indicating successful termination (ll. 28–30).

In normal operation, state consistency is satisfied by design since the decided state S_k is always reachable from the coordinator's previous state S_{k-1}. Incidentally, this is also the previous *decided* state if the coordinator decided in the previous period, or it is reachable from the last decided state since the coordinator always performs the state update in Algorithm 3, ll. 19–20 otherwise. Output consistency is satisfied since outputs are only generated upon deciding, and we assume that actuators apply output vectors labelled for period k only if received within the time interval $(t_{k-1}, t_k]$.

View Change Mode Normal operation is interrupted if at least one replica suspects the coordinator to have failed. Coordinator failures (real or suspected likewise) are handled by electing a different replica to be the coordinator and starting over in normal operation mode. As in [CT96; Dol+96], our algorithm uses a rotating coordinator approach, such that coordinators are chosen deterministically. To this end, a monotonically increasing view[6] number v is incremented upon suspected coordinator failure and used to agree upon the next coordinator $\texttt{coord}(v) = \left(v \bmod N\right) + 1$. This "view change" is performed in the procedure `NextView` shown in Algorithm 5, which is invoked either if no `Decide` message was received due to suspected coordinator failure (Algorithm 4, ll. 34, 36) or if a message with a higher view number was received (Algorithm 4, ll. 18–20, 32, 36). Note that messages with outdated view number are always discarded.

In normal operation, the persistent coordinator decides a single state for each period that is reachable from previously decided states. In contrast, when switching to a different coordinator, special care has to be taken that

(a) no two coordinators decide different states S_k for the same sampling period k (agreement), and

[6] While the term *round* is used in [CT96; Dol+96], we use the term *view* introduced in [OL88] because it is less likely to be confused with a *period*. The same concept is also referred to as *term* in [OO14].

Algorithm 5: View change executed on $r \in G$. Modifications w.r.t. [CT96; Dol+96] are shown in grey.

```
1  Function NextView(k, &est, &mode, &v, &v_b)
2      mode ← viewchange;
3      send ⟨Estimate, k, v, v_b, est⟩ to coord(v);
4      if r = coord(v) then
5          wait until received {⟨Estimate, k, v, v'_b, est'⟩} from
             G \ Suspects_r;
6          if received ⌈(N+1)/2⌉ Estimate messages then
7              select message with max(v'_b, est'.k_b);
8              est ← est';
9              v_b ← v'_b;
10         else if received message with v' > v then
11             v ← v';
12             NextView(k, &est, &mode, &v, &v_b);
13         else
14             v ← v + 1;
15             NextView(k, &est, &mode, &v, &v_b);
16         end
17     end
18     mode ← normal;
19     return;
20 end
```

(b) no coordinator proposes a state S_k that is not reachable from all states $S_{k'}$ decided in earlier periods $k' \leq k$.

The first concern is only relevant for view changes within the same consensus instance. The second concern is also relevant when proceeding from one *period* to the next, and therefore requires special treatment.

We first describe how conflicting decisions within the same sampling period are avoided (case *a*). While this is treated in [CT96; Dol+96], we briefly explain the mechanism here for the sake of completeness. Each replica only accepts messages with its own current view number v, and stores the view number of the last received `Propose` or `Decide` message as v_b (cf. Algorithm 4, ll. 15, 29). Because v is monotonically increasing on each replica, this is also the *most recent* view for which such a message was received. Therefore, if any coordinator decided a value in view v', then a majority of replicas must have received the corresponding `Propose` message and therefore have $v_b \geq v'$. Now, during view change all replicas (that suspected the old coordinator to have failed) send v_b together with their current estimate to the new coordinator (l. 3). The new coordinator must collect estimates for the new view from a majority of replicas (l. 6). This majority necessarily contains at least one witness to the most recent (possibly decided) proposal, which the new coordinator can identify by choosing a message with the largest v_b among those received (l. 7) and using the corresponding estimate for its next proposal (l. 8). Thereby, if a value was decided in the previous view, it is *impossible* for the new coordinator to decide a *different* value. If the new coordinator cannot gather a majority of estimates for its view, the next coordinator is chosen (ll. 14, 15).

While the agreement property is maintained by the consensus algorithm when coordinators change within the same period (i.e., consensus instance), we must also prevent newly elected coordinators from proposing any state that is not reachable from the decided states of previous periods (case *b*), which is crucial for maintaining state consistency. Since this introduces a dependency between subsequent consensus instances, we modified the view change mechanism to ensure that the new coordinator always selects an estimate containing a state that is valid w.r.t. state consistency. For this reason, each estimate contains the base period $\texttt{est}.k_b$ of the corresponding state, which is equal to the last received `Propose` or `Decide` message (cf. Algorithm 4, ll. 15, 29). Because the only way to modify a replica's state (S or $\texttt{est}.S$) without touching $\texttt{est}.k_b$ or v_b is through an `Update` in the main loop, and because k is monotonically increasing on each replica, we know that any controller's state estimate $\texttt{est}.S$ must be reachable from a state S' that was contained in an estimate proposed or decided in period $\texttt{est}.k_b$ and view v_b. Conversely, if any coordinator decided a state S' in period k' and view v', then a majority of replicas must have received the corresponding `Propose`

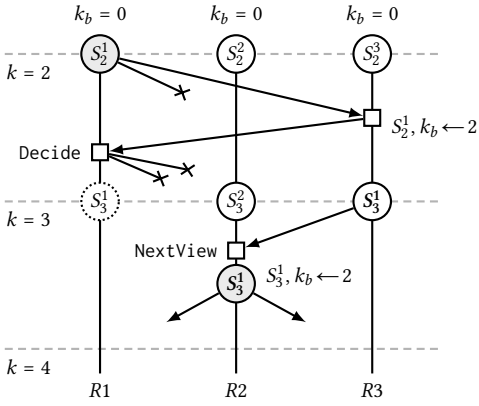


Figure 6.3: State-consistent viewchange example with three replicas. Coordinators are indicated by a shaded state node.

message and therefore have $\texttt{est}.k_b \geq k'$ and $v_b \geq v'$. The majority of estimates collected by the new coordinator in the `NextView` procedure necessarily contains at least one "witness" to the most recent (possibly decided) proposal. Now, the new coordinator can identify the most recent *possibly successful* consensus instance $C_{k'}$ by choosing the largest $\texttt{est}.k_b = k'$ *among* the received estimates with the most recent view[7] v_b (l. 7). If $C_{k'}$ was successful, the corresponding controller state $\texttt{est}.S$ is necessarily reachable from $S_{k'} = \texttt{Decision}(C_{k'})$. Finally, note that coordinators can only propose values as long as they are in normal operation mode. If a new coordinator was still waiting for estimates when the previous consensus instance was aborted (i.e., `mode` = viewchange at the beginning of the new period), it performs another view change (cf. Algorithm 4, ll. 3–6). Figure 6.3 shows a small example of how view change maintains state consistency.

Example 6.1. Figure 6.3 shows a group consisting of three replicas going into period $k = 2$ with different states and equal base period $k_b = 0$. Replica $R1$ (the coordinator) proposes its controller state S_2^1, which is acknowledged by replica $R3$ and subsequently decided. Therefore, $R1$ updates its proposed state ($S_2^1 \longrightarrow S_3^1$) and generates an output vector. Although $R1$'s `Decide` message to both participants is lost, replica $R3$ adopts the state (and base period) from the proposal for its next update, but does not generate an output. (Replica $R2$, not having received any message, simply performs a "local" update on its current state, also without generating an output.)

[7] Choosing an `est` with largest v_b is necessary for agreement [CT96].

In the next period $k = 3$, replica $R1$ becomes unavailable and replica $R2$ is elected as the new coordinator. Because $R2$ did not receive the proposal from $R1$ in $k = 2$, its current state S_3^2 is *not* reachable from the state S_2^1 decided in the previous period, and would therefore lead to a violation of state consistency if it were to be proposed for $k = 3$. However, as part of the election procedure it receives an estimate containing S_3^1 (which *is* reachable from S_2^1). Because this estimate has a higher base period ($k_b = 2$) than its own ($k_b = 0$), replica $R2$ adopts the state and proposes it for period $k = 3$. (Had $R2$ not received an estimate from either $R1$ or $R3$, it would not have procured the necessary majority to become coordinator and the next viewchange would have ensued.)

Note that `NextView` is aborted whenever the calling consensus instance is aborted, and that receiving a message with a higher view number within `NextView` triggers another viewchange (ll. 11, 12). Note also that viewchange is part of the single consensus loop in the original consensus algorithm, e.g., comprising Phase 1 and Phase 2 (except for the last line where the proposal is sent) in [CT96, Figure 6]. Our presentation of the algorithm is equivalent since every round of consensus with a different coordinator must be preceded by a completed view change, except for the initial round where the coordinator proposes its own estimate.

6.6.4 Correctness

Now we asses the presented algorithm with respect to the requirements laid down in Section 6.6.1. The correctness of individual consensus instances using Algorithms 4 and 5 with respect to *agreement*, *integrity*, and *termination* follows from the correctness of the original consensus algorithm which is proved in [CT96; Dol+96]. We note that we make only three essential modifications to the algorithm, none of which affect the properties of individual consensus instances. First, we add a period counter k to all relevant messages for distinguishing between different instances, which ensures that messages not belonging to the current consensus instance are rejected but does not alter the message pattern within the isolated instance. Second, we set the base period field $\texttt{est}.k_b$ of proposed estimates to k, which would be equivalent to setting the same part of all estimates to a fixed value in an isolated consensus instance. Third, we refine the selection of estimates during view change in Algorithm 5, l. 7 by adding a secondary criterion based on $\texttt{est}.k_b$. However, the chosen estimate is still from the set of messages with maximum v_b, and in the original algorithm an arbitrary estimate from this set would be selected.

The termination property may be violated because consensus instances are aborted in Algorithm 3 when the next period begins. But as shown in Section 6.6.1, we do not require the termination property to hold for individual consensus

instances. Indeed, consensus termination within one period cannot be guaranteed [FLP85; SWK09]. Moreover, eventual termination of each instance is not sufficient to argue about the availability of the replicated controller. However, our evaluation results in Section 6.8 show that our algorithm achieves a high availability in practice.

Theorem 6.2. *The protocol defined by Algorithms 3–5 satisfies Requirement 1, i.e.* (6.10) *and* (6.11).

Proof. Each consensus instance C_k executes the consensus algorithm from [CT96; Dol+96], which is possibly aborted prematurely (cf. Alg. 3, l. 17), and therefore satisfies the agreement property of consensus. Due to Alg. 3, ll. 22–24, all replicas that abort C_k do not send an output vector, i.e. $OV_k^r = \bot$. It follows that $\neg C_k \text{ successful} \implies \forall_{r \in G} OV_k^r = \bot$, which implies (6.11). Due to agreement, all replicas that do not abort yield the same state $S_k^r = \texttt{Decision}\big(C_k\big)$ and output $OV_k^r = \texttt{Output}(S)$, which implies (6.10). ■

Theorem 6.3. *The protocol defined by Algorithms 3–5 satisfies Requirement 2, i.e.* (6.12).

Proof. Assume C_k is successful and the controller state $S_k = \texttt{Update}\big(S_{k-1}, IV_{k-1}\big)$ is decided in view v. Therefore, a majority of replicas must have base period $k_b = k$, base view $v_b \geq v$, and controller state $S = \texttt{Decision}\big(C_k\big)$ at time t_k (cf. Alg. 3 after l. 21).

Now assume any replica r proposes an estimate with (S'_k, IV'_k) in view $v' \geq v$ for C_{k+1}. If $v' = v$, then $r = c$ and $S'_k = S_k$. If $v' > v$, then r must have chosen an estimate with S_k or (S_{k-1}, IV_{k-1}) in Alg. 5, which was acknowledged by a majority of replicas for C_k. Therefore, only an estimate with state S_k (and arbitrary IV) can be proposed for C_{k+1}, which is equivalent to the proposal of a state $S'_{k+1} \in \mathcal{R}\big(S_k\big)$ in terms of Property 2 and satisfies (6.12). By extension of the same argument, this is true for all $C_{k'}$, $k' > k$. ■

6.6.5 Discussion of the Algorithm

While we presented the SCRAM protocol in a way that is focused on reasoning about its correctness, it admits several optimizations for reducing the message overhead:

- For instance, the coordinator can omit the estimate from `Decide` messages to all replicas that acknowledged the corresponding `Propose`.
- Also, if the coordinator already decided for the previous period, it can omit the controller state from its estimate in `Propose` messages for the current period to replicas that acknowledged its `Propose` for the previous period.

- During viewchange, the `Estimate` messages can be reduced to contain only k, v, v_b, and k_b. The new coordinator can then request missing state information from only one replica.

While crash recovery was neither part of our system model nor considered in our description of the algorithm, replicas may also recover from crash failures. If a majority of replicas are always available, recovering replicas can listen for a `Propose` or `Decide` message and adopt the state information from the estimates contained therein, or they can receive `Estimate` messages from a majority of participants as part of normal leader election. After obtaining a valid state in that fashion, replicas can resume the algorithm as usual. However, if fewer than a majority of replicas may be available, they are required to log their state information in stable storage. Upon recovery, the controller state can be brought up to date (only in terms of the sampling period) by repeatedly applying `Update`$(S, [\perp, \perp, \ldots, \perp])$, with values for k_b and v_b remaining as prior to crash failure. Of course, the protocol can only make progress while a majority of replicas are available.

6.7 QoC-aware Replication

So far, we discussed the controller replication problem mainly in terms of consistency and availability. While increased availability can be expected to increase the closed-loop performance of the overall system, we now consider the impact of controller replication on QoC explicitly. Since QoC metrics are application-specific, and depend both on the characteristics of the physical system to be controlled and the controller itself, we perform a case study for a class of control systems that is specified in the following section.

6.7.1 LQG Control System Model

In the following, we consider control systems with a discrete-time LTI plant model and an observer-based output feedback controller.

The plant model is given by

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{6.13}$$

$$y_k = Cx_k + v_k, \tag{6.14}$$

where $x_k \in \mathbb{R}^{n_x}$ is the state, $u_k \in \mathbb{R}^{n_u}$ is the applied input, and $y_k \in \mathbb{R}^{n_y}$ is the measured output (which determines the controller's IV) of the plant at time t_k. The plant is affected by Gaussian disturbances through state noise $w_k \sim \mathcal{N}(0, W)$ and measurement noise $v_k \sim \mathcal{N}(0, V)$. We assume that (A, B) is controllable and (A, C) is observable.

To characterize the part of the measurements y_k that was successfully received by a controller, we introduce a binary arrival indicator vector $\gamma_k \in \{0,1\}^{n_y}$, where the sensor reading for components with $\gamma_k[i] = 0$ is not available to the controller. Correspondingly, the input vector IV_k is given as

$$IV_k[i] = \begin{cases} y_k[i] & \text{if } \gamma_k[i] = 1 \\ \perp & \text{if } \gamma_k[i] = 0. \end{cases} \tag{6.15}$$

The input applied to the plant is determined by the output vector (OV) of the replicated controller. However, since that value is undefined in sampling periods where the consensus instance is unsuccessful, cf. (6.11), we assume that the default applied input is zero, i.e.

$$u_k = \begin{cases} OV_k^r, r \in G_k^{\text{out}} & \text{if } C_k \text{ successful} \\ 0 & \text{otherwise,} \end{cases} \tag{6.16}$$

where we make the simplifying assumption that output vectors are delivered reliably to the actuators.

We consider an LQG optimal control set-up (cf. [Sin+05]) where the control objective is to minimize the expected infinite horizon average cost

$$J = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} x_k^\top Q x_k + 2x_k^\top H u_k + u_k^\top R u_k\right] \tag{6.17}$$

with weighting matrices Q, R, and H, such that $\left[\begin{smallmatrix} Q & H \\ H^\top & R \end{smallmatrix}\right] \succ 0$. Therefore, a low cost J corresponds to a high QoC.

The optimal controller consists of a Kalman filter to estimate the state x from the available measurements y and an LQR controller which uses the state estimate to calculate an optimal input u. The theoretically[8] optimal control u_k^* minimizing J is given by

$$u_k^* = K x_k \tag{6.18}$$

$$K = -(R + B^\top P B)^{-1} \cdot (B^\top P A + H^\top) \tag{6.19}$$

$$P = A^\top P A - (A^\top P B + H) \cdot (R + B^\top P B)^{-1} \cdot (B^\top P A + H^\top) + Q, \tag{6.20}$$

where, under the given assumptions, the algebraic Riccati equation (6.20) has a unique positive-definite solution P [DL71]. Note that both P and K are time-invariant and can be calculated off-line in advance.

As described above, the optimal control input u_k^* depends on the current state of the plant. However, the controller (replicas) cannot determine x_k exactly because

[8] since not in general realizable in our system model as the applied control input (6.16)

(a) a single measurement y_k is in general not sufficient to infer the plant state x_k (since C may not be invertible),

(b) measurements are affected by unknown noise v_k, and

(c) the input vectors containing these measurements—as received by the controller (replicas)—are affected by delay and loss.

Therefore, the controller can only approximate the optimal control signal based on the available information.

To this end, the controller employs a Kalman filter that maintains an estimate $\hat{x}$ of the plant's state and an estimate of the error covariance $\Sigma = \mathbb{E}\left[(\hat{x}_k - x_k) \cdot (\hat{x}_k - x_k)^\top\right]$. Since measurements may be partially missing in each IV, we consider the approach presented in [LG04] for Kalman filtering with partial observation losses in this case study. The rationale of that approach is to assume a measurement noise covariance approaching infinity for the missing components of y_k in the innovation step of the Kalman filter. In particular, let us assume that y_k (or an appropriate permutation of its components) is partitioned into $\left[y_k^{a\top}\ y_k^{d\top}\right]^\top$ such that $y_k^a \in \mathbb{R}^{|\gamma_k|}$ contains all available components of y_k and $y_k^d \in \mathbb{R}^{n_y - |\gamma_k|}$ contains all dropped measurement components (i.e., corresponding to $IV_k[i] = \perp$) in sampling period k. Considering the *a priori* measurement noise covariance matrix to be partitioned correspondingly as $V = \left[\begin{smallmatrix} V^a & V^{ad} \\ V^{da} & V^d \end{smallmatrix}\right]$, the block V^d is replaced in the innovation step for period k with σI, taking the limit $\sigma \longrightarrow \infty$.

Here, for ease of notation, we make the simplifying assumption that V is a diagonal matrix. Consequently, the state estimate $\hat{x}$ and error covariance Σ are updated in each period as

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \tag{6.21}$$

$$\Sigma_{k|k-1} = A\Sigma_{k-1|k-1}A^\top + W \tag{6.22}$$

$$L_k = \lim_{\sigma \to \infty} \Sigma_{k|k-1}C^\top\left(C\Sigma_{k|k-1}C^\top + V + \sigma \cdot (I - \operatorname{diag}(\gamma_k))\right)^{-1} \tag{6.23}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k(y_k - C\hat{x}_{k|k-1}) \tag{6.24}$$

$$\Sigma_{k|k} = (I - L_kC)\Sigma_{k+1|k}, \tag{6.25}$$

assuming that $x_0 \sim \mathcal{N}(\hat{x}_{0|0}, \Sigma_{0|0})$. As with the conventional Kalman filter, each iteration consists of a *prediction* step (6.21)–(6.22), where the state estimate is extrapolated using the plant model, followed by an *innovation* step (6.23)–(6.25), where measurement values from the input vector are incorporated to improve the accuracy of the state estimate.

Using the approach from [LG04] described earlier, the observer gain L_k is calculated in (6.23) with a term $\sigma \cdot (I - \operatorname{diag}(\gamma_k))$ added to the measurement noise

covariance V to account for missing values in the input vector IV_k. By letting $\sigma \longrightarrow \infty$, the result of the innovation step (6.24)–(6.25) becomes independent of the components of y_k with $\gamma_k[i] = 0$, i.e., those components which are missing in the received input vector IV_k. The limit of the matrix inverse in (6.23) can be calculated, for instance, using the Woodbury matrix identity. (Note that $L_k \longrightarrow 0$ if $\gamma_k = 0$, i.e., $\hat{x}$ and Σ remain unmodified in the innovation step when the IV is empty. It is also interesting to note that the error covariance $\Sigma_{k|k}$ does not converge to a stationary value for $k \longrightarrow \infty$ in general, since it depends on the arrival sequence γ.)

Algorithm 6: LQG `Update` function

Function `Update`(S, IV)
  $L \longleftarrow \lim_{\sigma \to \infty} S.\Sigma \cdot C^\top \left(C \cdot S.\Sigma \cdot C^\top + V + \sigma \cdot \mathrm{diag}([IV = \perp]) \right)^{-1}$;
  $\hat{x}^+ \longleftarrow A(S.\hat{x} + L(IV - C \cdot S.\hat{x})) + Bu$;
  $\Sigma^+ \longleftarrow A(I - LC) \cdot S.\Sigma \cdot A^\top + W$;
  return $(\hat{x}^+, \Sigma^+)$;
end

Algorithm 7: LQG `Output` function

Function `Output`(S)
  return $K \cdot S.\hat{x}$;
end

Algorithm 8: LQG `Cost` function

Function `Cost(est)`
  $\Sigma^+ \longleftarrow$ `Update`$\left(\mathtt{est}.S, \mathtt{est}.IV\right).\Sigma$;
  return $\mathrm{tr}\left(K^\top (B^\top P B + R) K \Sigma^+\right)$;
end

Since our system model contains a one-period delay in the controller execution model, i.e., the controller calculates a predictive OV for the following period, the controller performs the prediction step immediately after the innovation step. In terms of the generic controller model introduced in Section 6.3, the state $S_k = \left(\hat{x}_{k|k-1}, \Sigma_{k|k-1}\right)$ of the controller therefore consists of the Kalman filter's state

estimate and error covariance after the prediction step. The `Update` function (shown in Algorithm 6) takes the previous state S_{k-1} and last received input vector IV_{k-1} and applies equations (6.23) and (6.21) in line 3 for updating the state estimate and equations (6.25) and (6.22) in line 4 for updating the error covariance estimate, and returns the updated state S_k. The `Output` function (shown in Algorithm 7) takes the state S_k and produces the output vector $OV_k = K\hat{x}_{k|k-1}$.

Note that, for the sake of simplicity, we assume in Alg. 6, l. 3 that the update is performed with knowledge of the input u_{k-1} applied in the previous period, in order to maintain the separation principle. This assumption is somewhat realistic if OV messages are delivered with high reliability, which can also be increased by introducing acknowledgements from the actuators. (Replicas could receive these while waiting for IV and executing consensus, i.e., up to l. 19 in Alg. 3.) However, if this assumption cannot be realized, the controller has to be modified as in [Sin+08], for instance, which is often referred to as the "UDP-like case" in the literature.

6.7.2 Cost Model

Since the control cost J ultimately depends on the sequence of values produced by the replicated controller, we now investigate the influence of the underlying controller states.

Theorem 6.4. *Given an input sequence $(u_k)_{k=1}^{\infty}$, the expected cost is given by*

$$J = \operatorname{tr}(WP) + \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} \mathbb{E}\left[(u_k - Kx_k)^\top (B^\top PB + R)(u_k - Kx_k)\right] \tag{6.26}$$

Proof. We can rewrite equation (6.17) for the cost J by including a telescoping term in the sum as

$$J = \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} \mathbb{E}\left[x_k^\top Q x_k + 2x_k^\top H u_k + u_k^\top R u_k + x_{k+1}^\top P x_{k+1} - x_k^\top P x_k\right], \tag{6.27}$$

where we exploited the fact that $\lim_{T\to\infty} \frac{1}{T}\left(x_1^\top P x_1 - x_{T+1}^\top P x_{T+1}\right) = 0$. Note that

$$\begin{aligned}\mathbb{E}\left[x_{k+1}^\top P x_{k+1}\right] &= \mathbb{E}\left[(Ax_k + Bu_k + w_k)^\top P (Ax_k + Bu_k + w_k)\right] \\ &= \mathbb{E}\left[x_k^\top A^\top PAx_k + 2x_k^\top A^\top PBu_k + u_k^\top B^\top PBu_k + w_k^\top P w_k\right],\end{aligned}$$

where we used the independence of the zero-mean noise w_k. Inserting into (6.27)

and using the fact that $\mathbb{E}\left[w_k^\top P w_k\right] = \mathbb{E}\left[\mathrm{tr}(w_k w_k^\top P)\right] = \mathrm{tr}(WP)$ yields

$$J = \mathrm{tr}(WP) + \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} \mathbb{E}\left[x_k^\top (A^\top PA + Q - P)x_k + 2x_k^\top (A^\top PB + H)u_k + u_k^\top (B^\top PB + R)u_k\right]. \tag{6.28}$$

Finally, it follows from (6.19) and (6.20) that

$$K^\top (B^\top PB + R) = -(A^\top PB + H) \tag{6.29}$$

$$\text{and} \quad K^\top (B^\top PB + R)K = A^\top PA + Q - P, \tag{6.30}$$

and therefore (6.28) can be rewritten as

$$J = \mathrm{tr}(WP) + \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} \mathbb{E}\left[x_k^\top K^\top (B^\top PB + R)Kx_k - 2x_k^\top K^\top (B^\top PB + R)u_k + u_k^\top (B^\top PB + R)u_k\right],$$

which transforms directly into (6.26) through factorization. ■

From Theorem 6.4, it can be easily seen that $u_k^* = Kx_k$ is indeed the optimal control input. But moreover, it also shows the optimum achievable cost $\mathrm{tr}(WP)$ and to what degree each applied input u_k contributes to the total cost J. Remember that the applied input for the replicated controller depends on output vector OV_k and success of the corresponding consensus instance C_k as specified in (6.16).

6.7.3 Increasing QoC with SCRAM

If a single controller (6.18), (6.21)–(6.25) is used to control the plant (6.13)–(6.14), then the LQ cost J is determined by the sequence of measurement vector arrivals γ_k. However, the replicated controller offers an additional degree of freedom, since different replicas $r \in G$ may receive different input vector sequences, i.e., with different arrival patterns γ_k^r. In the nominal case (when the replicated controller is available), this degree of freedom vanishes since the coordinator "enforces" its own state on all influential replicas. But if consensus is unsuccessful for one or more sampling periods, the Kalman filters' state estimates of different replicas may diverge, leaving several options for determining the next output.

Let us consider the case that a consensus instance C_{k-1} terminates successfully on all replicas, leaving all in the same state $S_{k-1} = \texttt{Decision}\left(C_{k-1}\right)$. Now, if the instance C_k for the next sampling period fails with no replica receiving a `Propose`

message, and all replicas receive different input vectors $IV_k^r \neq IV_k^s \; \forall_{r \neq s}$, then in Alg. 3, l. 19 all replicas potentially obtain different states $S_k^r \neq S_k^s \; \forall_{r \neq s}$, but have the same view v, base view v_b, and base period $k_b = k - 1$. All replicas then enter the next consensus instance C_{k+1} in viewchange mode. If the new coordinator receives a majority of estimates in Alg. 5, l. 7, it chooses an arbitrary estimate, since all satisfy the requirements for state consistency. However, since the estimates contain different states, thus yielding different outputs u_{k+1}, the view change offers an opportunity for selecting the value which leads to the smallest stage cost, i.e., the smallest expected penalty on QoC.

Corollary. *Let the cost* (6.17) *be given in the form*

$$J = \lim_{T \to \infty} \frac{1}{T} \sum_{k=1}^{\infty} j_k.$$

The expected stage cost j_k incurred by a controller state $S_k = (\hat{x}_k, \Sigma_k)$ is given by

$$\operatorname{tr}(WP) + \operatorname{tr}\left(K^\top (B^\top P B + R) K \Sigma_k\right) \tag{6.31}$$

Proof. This follows from Theorem 6.4. Using (6.26), we can write the stage cost j_k as

$$\begin{aligned} j(u_k, x_k) &= \operatorname{tr}(WP) + \mathbb{E}\left[(u_k - K x_k)^\top (B^\top P B + R)(u_k - K x_k)\right] \\ &= \operatorname{tr}(WP) + \mathbb{E}\left[(\hat{x}_k - x_k)^\top K^\top (B^\top P B + R) K (\hat{x}_k - x_k)\right] \\ &= \operatorname{tr}(WP) + \operatorname{tr}\left(K^\top (B^\top P B + R) K \cdot \mathbb{E}\left[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^\top\right]\right) \\ &= \operatorname{tr}(WP) + \operatorname{tr}\left(K^\top (B^\top P B + R) K \Sigma\right) \end{aligned}$$

■

Based on Section 6.7.3, we can define a `Cost` function for consensus estimates, which is shown in Algorithm 8. Since the estimate for consensus instance C_k as defined in Algorithm 3 comprises S_{k-1} and IV_{k-1}, the method first calculates the updated state[9] and then returns the stage cost as defined in (6.31), discarding the constant term $\operatorname{tr}(WP)$.

Since the closed-loop cost J of the NCS is the average of the (non-negative) stage cost in each sampling period, we can improve QoC by minimizing the stage cost in every sampling period. Therefore, we propose two possible modifications of the viewchange mechanism that take the stage cost into account.

State-consistent QoC-based Selection Replacing line 7 in Algorithm 5 with

$$\text{select message with } \max\left(v_b', \texttt{est}'.k_b, -\texttt{Cost}(\texttt{est}')\right); \tag{6.32}$$

[9] In fact, it would be sufficient to update only the error covariance Σ.

maintains state consistency, since the consensus instance continues with an estimate with $\max(v'_b, \texttt{est}'.k_b)$ from a majority of replicas as in unmodified SCRAM, but selects the state incurring the smallest stage cost if there are multiple candidates.

Purely QoC-based Selection Replacing line 7 in Algorithm 5 with

$$\text{select message with } \max\left(v'_b, -\texttt{Cost}(\texttt{est}')\right); \tag{6.33}$$

maintains the agreement property (and thereby output consistency), since the consensus instance continues with an estimate with $\max v'_b$ from a majority of replicas as in the consensus algorithm by [CT96; Dol+96], but may violate state consistency, since a state which is not reachable from the latest influential state may be selected. However, the stage cost incurred by the selected state cannot be larger than in the state-consistent selection. Hence, this modification variant sacrifices the state-consistency property in favour of a potential further QoC improvement.

In Section 6.8.3, we will evaluate the QoC gains that can be achieved by these modifications with respect to the unmodified SCRAM algorithm using a simulation study of an inverted pendulum.

6.8 Evaluation

In this section, we evaluate the effectiveness and efficiency of SCRAM, comparing it to the existing real-time replication management protocol QUARTS for comparison. First, in Section 6.8.1 we evaluate the core performance metrics of the replication protocol, i.e., availability, latency, and message cost, without directly considering the effect on the closed-loop control system. To that end, we present evaluation results for an experiment with a physical plant in Section 6.8.2 for a control performance comparison. In Section 6.8.3, we then study the effect of the performance-optimizing protocol modifications proposed above.

6.8.1 Availability, Latency, Message Cost

The scenarios and failure model for our evaluation are based on those used in [Saa+17] for the evaluation of the QUARTS replication protocol in order to draw a fair comparison with that approach. We implemented both the QUARTS protocol and our own algorithm as discrete event simulations using the `SimJulia` simulation framework [Lau18; Bez+18] and were able to reproduce the results presented in [Saa+17] for the former.

Process failures obey the Gilbert-Elliot model which is often used to model bursty behaviour and consists of a two-state Markov chain with state-dependent failure probabilities. In any period, each replica may be either in a good (G) or bad (B) state. In the good state, the replica may be unavailable for the current period according to a Bernoulli process with probability θ_d to model sporadic failures (which are attributed to erratic computation delay in [Saa+17]). In the bad state, the replica remains unavailable until it returns to the good state in a future period, to model crash failures. The stationary probability of being in the crashed (bad) state is given by θ_c while the mean time to repair (MTTR) after which replicas recover crash failures on average is given by R. This implies that the transition probability for B⟶G (recover) equals $\frac{T_s}{R}$, while that for G⟶B (crash) equals $\frac{T_s\theta_c}{R(1-\theta_c)}$. Because the number of concurrently crashed replicas cannot be bounded in this model, we simulated a recovery operation based on the assumption of stable storage as outlined in Section 6.6.5.

Communication is modelled according to the so-called *probabilistic synchronous* [Dzu+16], which is also used in [Saa+17]. Message loss follows a Bernoulli process with probability p, while message delay is governed by an i.i.d. process with uniform distribution on $(0, \delta]$. The failure detector used in the simulation of our protocol is implemented as a message reception timeout based on the network delay δ.

For all simulations, we take

- the number of sensor nodes $n_y = 10$,
- the sampling interval $T_s = 20\,\text{ms}$,
- the network delay $\delta = 0.5\,\text{ms}$,
- the message loss probability $p = 1 \times 10^{-3}$,
- the MTTR for crash failures $R = 1\,\text{s}$,
- the sporadic failure probability $\theta_d = 1 \times 10^{-3}$, and
- the crash failure probability $\theta_c = 1 \times 10^{-4}$,

as in [Saa+17] unless noted otherwise, with $N = 3$ replicas.

We performed three parameter studies, varying the message loss rate $p \in [10^{-4}, 10^{-1}]$, the crash failure rate $\theta_c \in [10^{-4}, 10^{-1}]$, and the sampling interval $T_s \in [1\,\text{ms}, 20\,\text{ms}]$, respectively. For each parameter study, we ran simulation experiments for 900 000 sampling periods, which corresponds to a simulated time of five hours.

The results for varying message loss rate are shown in Figure 6.4. The top pane shows the unavailability of both approaches. While our algorithm outperforms

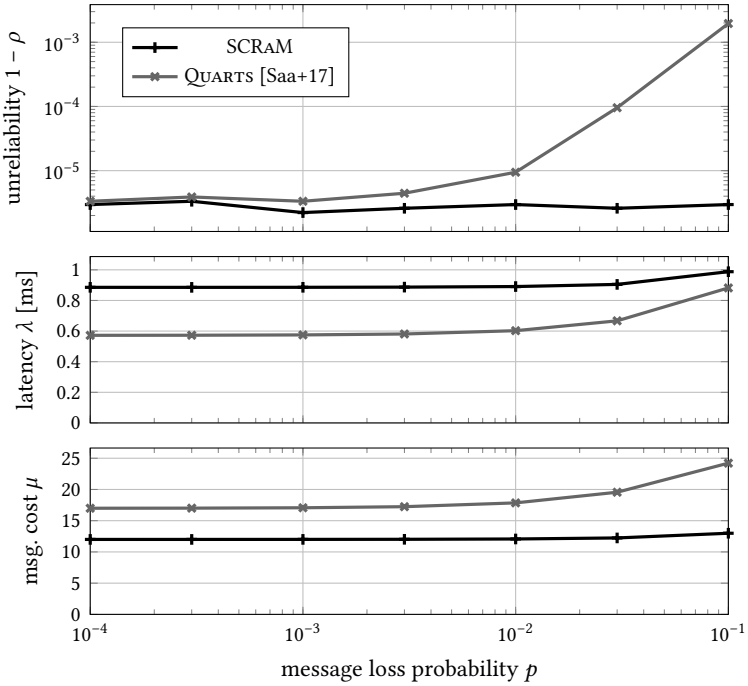


Figure 6.4: Parameter study for $p \in [10^{-4}, 10^{-1}]$

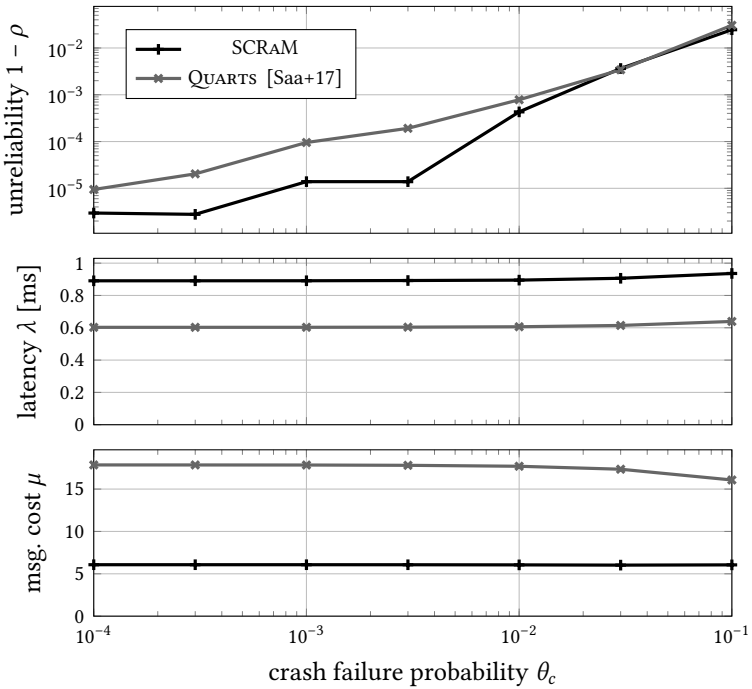


Figure 6.5: Parameter study for $\theta_c \in [10^{-4}, 10^{-1}], p = 10^{-2}$

Quarts throughout this parameter range, we can see that both approaches perform similarly at low message loss rates. However, our approach remains robust to increasing p while the unavailability of Quarts increases by approximately three orders of magnitude as p increases by the same order. The middle pane shows the mean latency λ. Here we can observe that Quarts outperforms our algorithm by up to 55 %. However, the bottom pane shows that the lower latency of Quarts comes at a cost of more than twice the message cost compared to our algorithm. Latency and messaging cost increase for both algorithms in response to inceased loss probability, although our algorithm can be seen to be less sensitive to p with respect to both metrics.

The results for varying crash failure rate are shown in Figure 6.5, where the message loss rate is fixed at $p = 10^{-2}$. Again, the top pane shows the unavailability of both approaches, which indicates that our algorithm outperforms Quarts by approximately an order of magnitude for $\theta_c < 10^{-2}$, while the performance of both approaches converges for larger crash failure rates. (We note that this

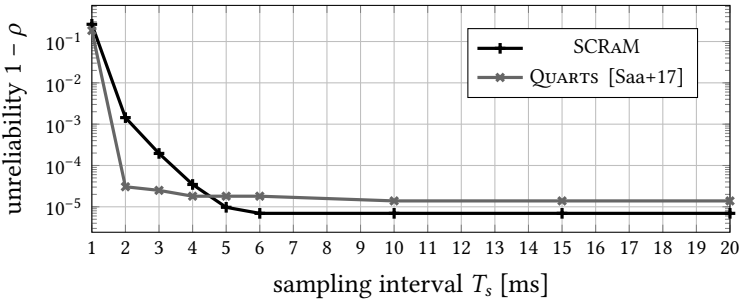


Figure 6.6: Parameter study for $T_s \in [1\,\text{ms}, 20\,\text{ms}]$

effect becomes more pronounced for larger p and less pronounced for smaller p, and that we chose $p = 10^{-2}$ to make the qualitative properties visible. For vanishing p we observed that both algorithms have the same availability, which indicates that our algorithm is more robust overall to message loss.) With respect to latency (middle pane), we see a similar difference between both algorithms as in the previous scenario, while λ remains almost constant with respect to θ_c. The messaging cost (bottom pane) also shows a similar ratio, but decreases slightly for Quarts at higher crash failure rates.

The higher average latency of our algorithm raises the expectation that its advantages should diminish as the available timespan for successfully executing consensus, i.e., the sampling interval T_s, decreases. The unavailability of both algorithms is compared for varying T_s in Figure 6.6. Indeed, it shows that our algorithm is outperformed by Quarts for $T_s < 5\,\text{ms}$. Since the latency of Quarts is bounded by $6\delta = 3\,\text{ms}$ (5δ for the protocol [Saa+17] plus δ for receiving IV), it is only below this value that its availability significantly deteriorates. By contrast, our algorithm makes better use of the available "slack time" at lower message cost by proceeding in rounds (i.e., views) up to T_s, but has higher mean and tail latency. This indicates that the choice of replication algorithm (leaving state consistency aside) must be informed by the sampling period in relation to the network delay.

6.8.2 NCS Performance

In order to assess SCRaM with respect to QoC, we also evaluated it using an experimental CPS set-up with a physical plant. For these experiments, we increased the crash failure probability to $\theta_c = 0.1$ and the MTTR to $R = 3\,\text{s}$.

The physical part of the experiment is an inverted pendulum (whose angle we denote by θ) balancing on a cart (whose position we denote as x). The cart is

driven by a stepper motor using a belt assembly. A microcontroller drives the stepper, reads θ using an incremental rotary encoder, and communicates with the PC running the event-based simulation of the replicated controller, which is periodically suspended until receiving a new measurement, to keep in sync with real time.

We used the LQG design shown in Section 6.7 for the $N = 3$ controller replicas, i.e., `Update` is one iteration of a Kalman filter recursion, while `Output` applies an LQR gain to the filter's state estimate, as shown in Algorithms 6 and 7. The sampling period is $T_s = 50\,\text{ms}$. The cart-driven inverted pendulum that we used in our experiments is shown in Figure 6.7. A rod of 0.6 m length is mounted on the shaft of an incremental rotary encoder with $600\frac{\text{pulse}}{\text{rev}}$. The track has a usable range of 1.2 m.

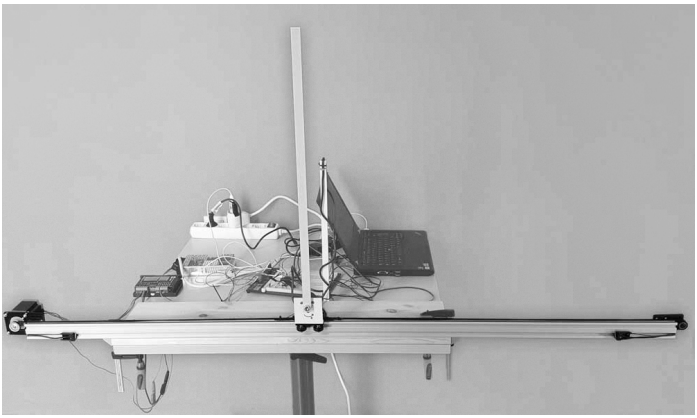


Figure 6.7: Physical inverted pendulum for replication evaluation

We denote the state of the system by $\xi = [x, \dot{x}, \theta, \dot{\theta}]^\top$, where x is the distance of the cart from the origin (track center) in meters and θ is the angle of the pole from upright in radians. Sensor measurements are $y = [x, \theta]^\top$ and the actuation input is the cart acceleration $u = \ddot{x}$. For the design of the LQG controller, we use the continuous-time cost metric

$$J = \int_{t=0}^{\infty} \xi^\top Q_c\, \xi + u^\top R_c\, u \;\mathrm{d}t$$

with $Q = \operatorname{diag}\big([1, 0.01, 1.5, 0.01]\big)$ and $R = 0.02$. Linearization and discretization at

$T_s = 50\,\text{ms}$ of the non-linear pendulum equation gives a discrete-time LTI system

$$\xi_{k+1} = A\xi_k + Bu_k + w_k \tag{6.34}$$

$$y_k = C\xi_k + v_k \tag{6.35}$$

$$J_T = \frac{1}{T}\sum_{k=0}^{T} \xi_k^\top Q\xi_k + 2\xi_k^\top H u_k + Ru_k^2 \tag{6.36}$$

with

$$A = \begin{bmatrix} 1 & 0.05 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1.018 & 0.05 \\ 0 & 0 & 0.705 & 1.018 \end{bmatrix} \quad B = \begin{bmatrix} 0.125 \\ 5.0 \\ 0.179 \\ 7.185 \end{bmatrix} \cdot 10^{-2} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.37}$$

$$Q = \begin{bmatrix} 5.0 & 0.125 & 0 & 0 \\ 0.125 & 0.054 & 0 & 0 \\ 0 & 0 & 7.596 & 0.207 \\ 0 & 0 & 0.207 & 0.057 \end{bmatrix} \cdot 10^{-2} \quad H = \begin{bmatrix} 2.083 \\ 1.328 \\ 5.359 \\ 1.975 \end{bmatrix} \cdot 10^{-5} \quad R = 1.001 \times 10^{-3} \tag{6.38}$$

The LQR optimal input is

$$u_k = \begin{bmatrix} 5.295 & 5.967 & -42.519 & -11.239 \end{bmatrix} \cdot \xi_k. \tag{6.39}$$

As noise covariance matrices for w_k and v_k we take

$$W = \begin{bmatrix} 5.042 \times 10^{-5} & 1.25 \times 10^{-5} & 0 & 0 \\ 1.25 \times 10^{-5} & 5 \times 10^{-4} & 0 & 0 \\ 0 & 0 & 5.143 \times 10^{-3} & 0.004 \\ 0 & 0 & 4.301 \times 10^{-3} & 0.102 \end{bmatrix} \tag{6.40}$$

and

$$V = \begin{bmatrix} 1.5 \times 10^{-4} & 0 \\ 0 & 2.5 \times 10^{-3} \end{bmatrix} \tag{6.41}$$

Our experiment consisted of balancing the pendulum for 180 s, starting upright with the cart at the origin. We repeated the experiment 25 times for each protocol, always with the same realization (seed value) of the crash failure model. The average availability—i.e., the proportion of periods with $OV \neq \perp$—was 99.2 % for all experiments. In Figure 6.8, the diagrams in the top and middle show the cumulative distribution of the maximum absolute pole angle and the position range covered by the cart per experiment. Even though both replication protocols execute the same controller function and achieve the same availability, the NCS

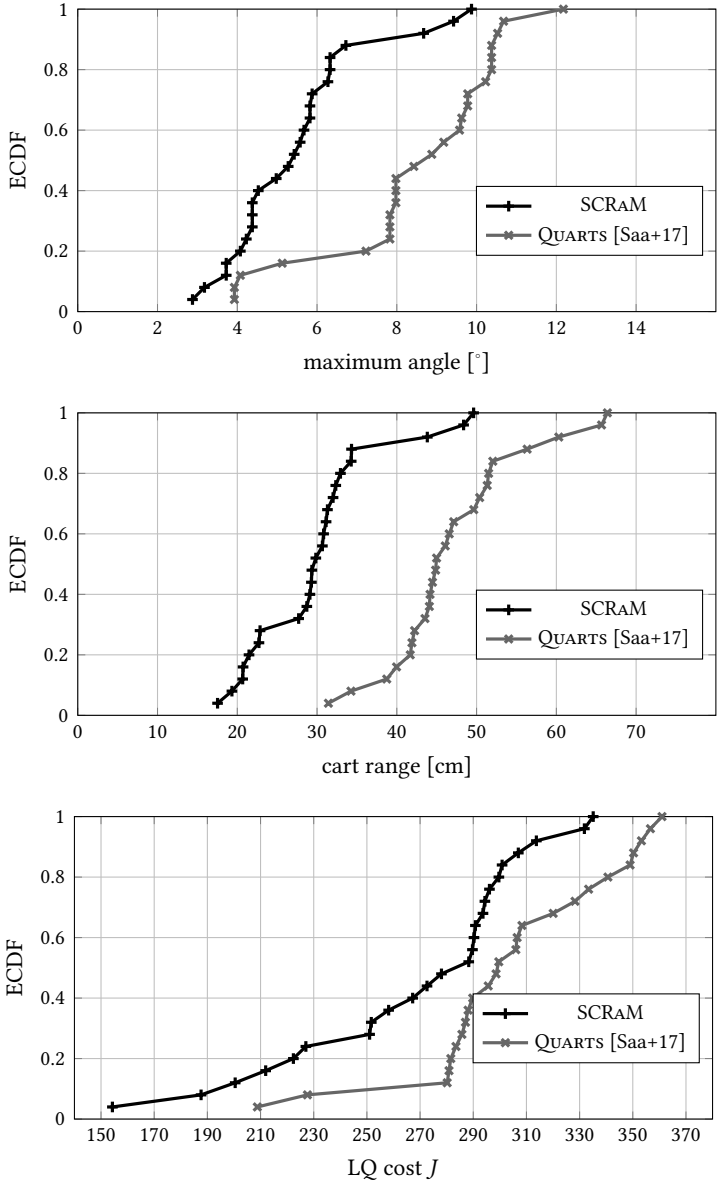


Figure 6.8: Empirical CDFs over all experiments for both replication protocols: maximum absolute pendulum angle $\max_k |\theta_k|$ (top), cart position range $\max_k x_k - \min_k x_k$ (middle), LQ cost J (bottom)

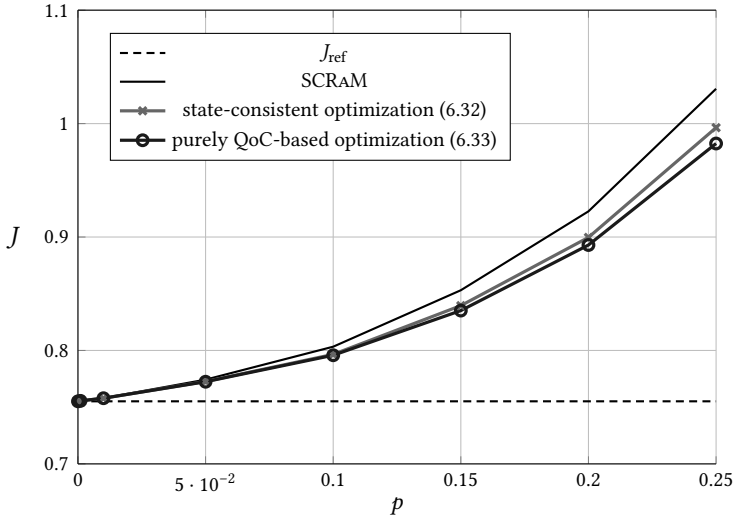


Figure 6.9: Comparison of LQ cost J using the unmodified and modified SCRaM protocol for $p \in [10^{-5}, 2.5 \cdot 10^{-2}]$

replicated using SCRaM keeps the angle 2.97° (or 35 %) smaller and the cart range 17.15 cm (or 36 %) smaller on average than with Quarts. This demonstrates that providing state consistency with SCRaM can offer a clear performance advantage. The bottom diagram shows the empirical CDF of the LQ cost J for all experiments, which on average is 14.8 % larger when using Quarts compared to SCRaM.

6.8.3 QoC Optimization

Finally, we assess the QoC optimizing protocol modifications proposed in Section 6.7.3. To this end, we simulated the linearized pendulum model (6.34)–(6.41) as the plant, using the same controller as in the experiments in Section 6.8.2. Our experiments for this evaluation consisted of balancing the simulated pendulum for 600 s. We kept the sampling period $T_s = 50$ ms as in the previous section, and set the remaining parameters $\delta = 0.5$ ms, $R = 1$ s, $\theta_d = 1 \times 10^{-3}$, and $\theta_c = 1 \times 10^{-4}$ as in Section 6.8.1. Each experiment was repeated ten times with different seed values for the random failures and noise.

In Figure 6.9, the average LQ cost of the experiments is shown for the unmodified SCRaM protocol and the two proposed modifications over varying packet loss probability $p \in [10^{-5}, 2.5 \cdot 10^{-2}]$. For reference, the horizontal dashed line indicates the average cost $J_{\text{ref}} \approx 0.755$ for corresponding experiments with zero

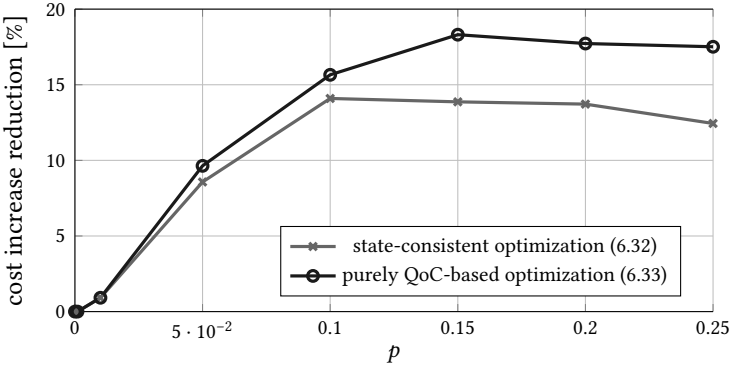


Figure 6.10: Comparison of LQ cost increase $J - J_{\text{ref}}$ using modified protocols relative to cost increase of unmodified SCRaM protocol for $p \in [10^{-5}, 2.5 \cdot 10^{-2}]$

packet loss and crash probability $p = \theta_d = \theta_c = 0$. This is the fundamental baseline cost of the optimal failure-free centralized controller.

We can see that for a low packet loss probability, the cost J_{SCRAM} of the unmodified protocol is already very close to the optimal cost J_{ref}. Specifically, the average cost J_{SCRAM} for $p = 10^{-5}$ is within $\left(1 + 6.4 \cdot 10^{-8}\right) \cdot J_{\text{ref}}$, and therefore no significant potential for performance optimization is given in this regime. Of course, when the packet loss rate is increased, the closed-loop performance deteriorates correspondingly, due to both intermittent unavailability of the replicated controller and loss of sensor measurements. However, we can see that the protocol modification with the *state-consistent QoC-based selection* (6.32) consistently yields a lower LQ cost. Moreover, the *purely QoC-based selection* (6.33) offers only a marginal cost improvement over the state-consistent method.

In order to better quantify this effect, we consider the packet–loss–induced cost increase $J - J_{\text{ref}}$ with respect to the optimum achievable cost. Figure 6.10 shows the percentage by which the two proposed protocol modifications reduce the cost increase beyond J_{ref} compared to the unmodified protocol, which is given by

$$\frac{J_{\text{SCRAM}} - J}{J_{\text{SCRAM}} - J_{\text{ref}}}.$$

We can see that an application-specific, performance-aware selection of estimates in the viewchange mode of the replication protocol can reduce the cost increase induced by packet losses by up to 14.1 % while maintaining state consistency, by using the first proposed protocol modification (6.32). Moreover, a cost increase

reduction of up to 18.3 % is possible if state consistency is abandoned, by using the second proposed protocol modification (6.33).

We note that the same experiments using the QUARTS protocol yielded diverging closed-loop behaviour (and cost) for packet loss probabilities beyond $p = 5 \cdot 10^{-3}$, and are therefore not shown in the plot. Therefore, our evaluation validates state consistency as the most useful performance-maintaining mechanism for controller replication. While packet–loss–induced performance degradation can be mitigated to a certain degree by introducing application-specific modifications into the replication management protocol, further optimizations that violate state consistency only offer marginal performance gains, and are of interest mainly for very high packet loss regimes.

6.9 Summary and Outlook

In this chapter, we defined two consistency concepts for controller replication in NCS, namely *output* and *state consistency*, which together ensure that a replicated controller can be used as a functionally indistinguishable drop-in replacement for a non-replicated controller. With SCRAM, we presented a corresponding efficient replication protocol. Our evaluations showed that it outperforms the state of the art for NCS controller replication in terms of both reliability and message cost over a wide range of parameters, and indicate in which cases our algorithm is the most suitable.

While SCRAM can be used very generically for periodically sampled controllers and is agnostic to the underlying control design methodology, we showed how control-specific performance metrics can be used to inform the choice of states to propose by modifying the view change algorithm for a special case of LQG control.

One possible avenue for future work is the investigation of different consensus protocols in SCRAM. For instance, using the family of protocols recently described in [HM19] may allow us to investigate latency–reliability trade-offs.

7 Summary

NCS can be found in diverse application areas, including automotive and aircraft systems, smart homes, and smart manufacturing systems in the context of Industry 4.0. While contemporary NCS often use general-purpose networks simply as a more cost-effective, flexible, and/or maintainable alternative for specialized field-bus networks or extensive dedicated wiring, the trend of steadily increasing digitization will likely lead to applications with a higher level of system integration, especially in large-scale systems such as smart grids and smart cities. This leads to major challenges concerning the provisioning of appropriate communication services, due to the high QoS requirements of control systems, but also—and more fundamentally—due to the current shortage of compatibility between control and communication with respect to models and methods. On the one hand, while NCS have been studied intensively from the control perspective over the last decades, the design and analysis methods in those studies have been based on simplistic and abstract network connection models. On the other hand, communication networks are optimized for conventional performance metrics such as throughput and latency, which do not readily translate into application specific control performance metrics. This division hinders the holistic co-design of complex NCS that is required for the novel applications outlined above. In this thesis, we have aimed to reduce that gap by providing performance-oriented communication concepts for NCS. As such, the main contributions of this thesis are:

- In Chapter 3, we have addressed the scheduling problem for a group of NCS sharing a dedicated network slice by designing a performance-aware dynamic priority scheduler. The scheduler uses state-based packet priorities calculated at the sensors, which are then used for stateless priority queuing in the network, making it both scalable and efficient to implement at the data-link layer. The parameters for calculating the packet priorities are determined by solving a global SDP optimization problem for the group. By incorporating LMI stability constraints for switched systems, the resulting scheduling policy provides asymptotic stability guarantees for each NCS and performance bounds on the joint QoC. Evaluation with a proof-of-concept implementation has shown a 68% LQ cost reduction compared to round-robin scheduling for a group of 6 simulated inverted pendulums.

- In Chapter 4, we have designed a state-based priority scheduler for opportunistic traffic in the deterministic–opportunistic transmission slot (DOTS) model. We have decoupled the stability guarantees for individual NCS, which are provided by deterministic transmissions, from the opportunistic joint QoC optimization for the group, which is implemented by the priority scheduler. The resulting scheduler requires no QoS guarantees for the opportunistic transmissions and provides worst-case bounds on the joint QoC. In contrast to the scheduler developed in Chapter 3, which requires a comparatively large-scale SDP to be solved, the opportunistic scheduler only requires the solution of a comparatively small algebraic matrix equation for each NCS, and supports incremental addition and removal of NCS from the group of scheduled applications. Moreover, a co-design method for a time-varying controller has been provided, which optimizes the joint QoC under worst case assumptions. Numerical evaluation has shown that the achieved QoC is comparable to that of the first scheduler, while calculation of the priority parameters is much faster (by a factor of ≈300) and stability analysis is far less conservative.
- In Chapter 5, we have developed a cross-layer communication service for NCS with a probabilistic Bernoulli packet loss model and a corresponding routing algorithm for finding QoC-optimal paths in a network with stochastic link delays. The approach is based on a QoS specification that expresses the minimal required in-time arrival probability as a function of the sampling period. The routing algorithm solves the constrained graph optimization problem of finding a feasible path that admits a maximum sampling period through dynamic programming. Numerical evaluation has shown that the cross-layer service is effective at maintaining target QoC. For a simple test scenario, routing offered a 6% network load improvement compared to mere sampling rate adaptation on a feasible path.
- In Chapter 6, we have addressed the problem of active replication for controllers. To this end, we have developed the state consistency condition that provides replication transparency for control systems. This abstraction allows the design of the control law for a replicated controller to be treated the same as for a non-replicated controller. We have also presented the corresponding replication management protocol SCRAM, and have shown in our evaluation that it achieves high availability and low latency at low message cost compared to existing real-time replication protocols that only provide output consistency. Moreover, we have studied how QoC degradation due to failures can be mitigated in active replication by incorporating application-specific performance metrics into the replication management protocol.

In conclusion, this thesis provides control-specific scheduling, routing, and replication methods on the basis of integrated models of control and communication, which enables the provisioning of performance-oriented communication services for NCS and allows their design goals to be specified in terms of closed-loop QoC.

Bibliography

[ÅB02] K. J. Åström and B. M. Bernhardsson. "Comparison of Riemann and Lebesgue sampling for first order stochastic systems". In: *Proceedings of the 41st IEEE Conference on Decision and Control*. CDC. Las Vegas, Nevada, USA, Dec. 2002, pp. 2011–2016. DOI: 10.1109/CDC.2002.1184824.

[ADK14] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers. "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks". In: *IEEE Network Operations and Management Symposium (NOMS)*. May 2014, pp. 1–8. DOI: 10.1109/NOMS.2014.6838228.

[AGL15] S. Al-Areqi, D. Görges, and S. Liu. "Event-based networked control and scheduling codesign with guaranteed performance". In: *Automatica* 57 (July 2015), pp. 128–134. DOI: 10.1016/j.automatica.2015.04.003.

[Al-+13] S. Al-Areqi, D. Görges, S. Reimann, and S. Liu. "Event-based control and scheduling codesign of networked embedded control systems". In: *Proceedings of the American Control Conference (ACC)*. Washington, DC, USA, June 2013, pp. 5299–5304. DOI: 10.1109/ACC.2013.6580665.

[AR15] F. Allgöwer and K. Rothermel. *Integrated Controller Design Methods and Communication Services for Networked Control Systems (NCS)*. Research Grant – Deutsche Forschungsgemeinschaft (DFG). 2015. URL: https://gepris.dfg.de/gepris/projekt/285825138.

[BA11] R. Blind and F. Allgöwer. "Analysis of Networked Event-Based Control with a Shared Communication Medium: Part I – Pure ALOHA". In: *Proceedings of the 18th IFAC World Congress*. Aug. 2011, pp. 10092–10097. DOI: 10.3182/20110828-6-IT-1002.01100.

[BA12] R. Blind and F. Allgöwer. "Is it worth to retransmit lost packets in Networked Control Systems?" In: *Proceedings of the 51st IEEE Conference on Decision and Control*. CDC. Maui, HI, USA, Dec. 2012, pp. 1368–1373. DOI: 10.1109/CDC.2012.6426881.

[BA13] R. Blind and F. Allgöwer. "On the Optimization of the Transport Layer for Networked Control Systems". In: *at – Automatisierungstechnik* 61.7 (July 2013), pp. 495–505. DOI: 10.1524/auto.2013.1028.

[BA14] R. Blind and F. Allgöwer. "On the stabilizability of continuous-time systems over a packet based communication system with loss and delay". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 6466–6471. ISSN: 1474-6670. DOI: 10.3182/20140824-6-ZA-1003.01213.

[BA15] R. Blind and F. Allgöwer. "Towards Networked Control Systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process". In: *Proceedings of the 54th IEEE Conference on Decision and Control*. CDC. Osaka, Japan, Dec. 2015, pp. 7510–7515. DOI: 10.1109/CDC.2015.7403405.

[Bak+01] F. Baker, C. Iturralde, F. L. Faucheur, and B. Davie. *Aggregation of RSVP for IPv4 and IPv6 Reservations*. RFC 3175. IETF, Sept. 2001. URL: http://tools.ietf.org/rfc/rfc3175.txt.

[Bal+18] A. Ballesteros, J. Proenza, M. Barranco, and L. Almeida. "Reconfiguration Strategies for Critical Adaptive Distributed Embedded Systems". In: *Proceedings of the 48th IEEE/IFIP International Conference on Dependable Systems and Networks Workshops*. June 2018, pp. 57–58. DOI: 10.1109/DSN-W.2018.00028.

[Bar+17] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, and V. Verdugo. "A Scheduling Model Inspired by Control Theory". In: *Proceedings of the 25th International Conference on Real-Time Networks and Systems*. RTNS '17. Grenoble, France: ACM, 2017, pp. 78–87. ISBN: 978-1-4503-5286-4. DOI: 10.1145/3139258.3139272.

[Ben+12] J. W. Bennett, G. J. Atkinson, B. C. Mecrow, and D. J. Atkinson. "Fault-Tolerant Design Considerations and Control Strategies for Aerospace Drives". In: *IEEE Transactions on Industrial Electronics* 59.5 (May 2012), pp. 2049–2058. DOI: 10.1109/TIE.2011.2159356.

[Bez+18] J. Bezanson, J. Chen, B. Chung, S. Karpinski, V. B. Shah, J. Vitek, and L. Zoubritzky. "Julia: Dynamism and Performance Reconciled by Design". In: *Proceedings of the ACM on Programming Languages* 2.OOPSLA (Oct. 2018). ISSN: 2475-1421. DOI: 10.1145/3276490.

[BHJ10] A. Bemporad, W. P. M. H. M. Heemels, and M. Johansson, eds. *Networked Control Systems*. Springer, 2010. DOI: 10.1007/978-0-85729-033-5.

[BK14] P. Bailis and K. Kingsbury. “The Network is Reliable”. In: *Communications of the ACM* 57.9 (Sept. 2014), pp. 48–55. DOI: 10.1145/2643130.

[BSS08] M. E. M. Ben Gaïd, D. Simon, and O. Sename. “A Design Methodology for Weakly-Hard Real-Time Control”. In: *Proceedings of the 17th IFAC World Congress*. IFAC WC ’08 inria-00269209. IFAC. Seoul, South Korea, July 2008, p. 7. URL: https://hal.inria.fr/inria-00269209.

[Car+12] B. W. Carabelli, A. Benzing, G. Seyboth, R. Blind, M. Bürger, F. Dürr, B. Koldehofe, K. Rothermel, and F. Allgöwer. “Exact Convex Formulations of Network-Oriented Optimal Operator Placement”. In: *Proceedings of the 51st IEEE Conference on Decision and Control* (Maui, HI, USA). CDC. IEEE, Dec. 2012, pp. 3777–3782. DOI: 10.1109/CDC.2012.6426790.

[Car+17] B. W. Carabelli, R. Blind, F. Dürr, and K. Rothermel. “State-Dependent Priority Scheduling for Networked Control Systems”. In: *Proceedings of the 2017 American Control Conference* (Seattle, WA, USA). ACC. IEEE, May 2017, pp. 1003–1010. DOI: 10.23919/ACC.2017.7963084.

[CB10] N. M. K. Chowdhury and R. Boutaba. “A survey of network virtualization”. In: *Computer Networks* 54.5 (Apr. 2010), pp. 862–876. DOI: 10.1016/j.comnet.2009.10.017.

[CDK01] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design*. 3rd ed. Addison-Wesley, 2001. ISBN: 0201619180.

[CDR20] B. W. Carabelli, F. Dürr, and K. Rothermel. “SCRaM – State-Consistent Replication Management for Networked Control Systems”. In: *Proceedings of the 11th IEEE/ACM International Conference on Cyber-Physical Systems* (Sydney, NSW, Australia). ICCPS. IEEE, Apr. 2020. DOI: 10.1109/ICCPS48487.2020.00035.

[CF99] F. Cristian and C. Fetzer. “The Timed Asynchronous Distributed System Model”. In: *IEEE Transactions on Parallel and Distributed Systems* 10.6 (June 1999), pp. 642–657. DOI: 10.1109/71.774912.

[Ché+92] M. Chérèque, D. Powell, P. Reynier, J.-L. Richier, and J. Voiron. “Active replication in Delta-4”. In: *Proceedings of the 22nd International Symposium on Fault-Tolerant Computing*. July 1992, pp. 28–37. DOI: 10.1109/FTCS.1992.243618.

[Clo+06] M. Cloosterman, N. van de Wouw, M. Heemels, and H. Nijmeijer. “Robust Stability of Networked Control Systems with Time-Varying Network-Induced Delays”. In: *Proceedings of the 45th IEEE Conference Decision and Control*. CDC. San Diego, CA, USA: IEEE, Dec. 2006, pp. 4980–4985. DOI: 10.1109/CDC.2006.376765.

[CMF05] O. L. V. do Costa, R. P. Marques, and M. D. Fragoso. *Discrete-Time Markov Jump Linear Systems*. Springer, 2005. DOI: 10.1007/b138575.

[Cra+16] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner. "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks". In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France: ACM, 2016, pp. 183–192. ISBN: 978-1-4503-4787-7. DOI: 10.1145/2997465.2997470.

[Cro96] J. Crowcroft. *Open Distributed Systems*. Artech House, 1996. ISBN: 0890068399.

[CT96] T. D. Chandra and S. Toueg. "Unreliable Failure Detectors for Reliable Distributed Systems". In: *Journal of the ACM* 43.2 (Mar. 1996), pp. 225–267. ISSN: 0004-5411. DOI: 10.1145/226643.226647.

[DHL15] I. Dunning, J. Huchette, and M. Lubin. "JuMP: A modeling language for mathematical optimization". In: (2015). arXiv: 1508.01982 [math.OC].

[DL71] P. Dorato and A. H. Levis. "Optimal linear regulators: The discrete-time case". In: *IEEE Transactions on Automatic Control* 16.6 (Dec. 1971), pp. 613–620. DOI: 10.1109/TAC.1971.1099832.

[DLG10] S.-L. Dai, H. Lin, and S. S. Ge. "Scheduling-and-Control Codesign for a Collection of Networked Control Systems With Uncertain Delays". In: *IEEE Transactions on Control Systems Technology* 18.1 (Jan. 2010), pp. 66–78. ISSN: 1063-6536. DOI: 10.1109/TCST.2008.2010459.

[DN16] F. Dürr and N. G. Nayak. "No-Wait Packet Scheduling for IEEE Time-Sensitive Networks (TSN)". In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France: ACM, 2016, pp. 203–212. ISBN: 978-1-4503-4787-7. DOI: 10.1145/2997465.2997494.

[Doh+04] J. L. Dohner, J. P. Lauffer, T. D. Hinnerichs, N. Shankar, M. Regelbrugge, C.-M. Kwan, R. Xu, B. Winterbauer, and K. Bridger. "Mitigation of chatter instabilities in milling by active structural control". In: *Journal of Sound and Vibration* 269.1 (2004), pp. 197–211. ISSN: 0022-460X. DOI: 10.1016/S0022-460X(03)00069-5.

[Dol+96] D. Dolev, R. Friedman, I. Keidar, and D. Malkhi. *Failure Detectors in Omission Failure Environments*. Tech. rep. 1813/7263. Ithaca, NY, USA: Cornell University, Sept. 1996. DOI: 1813/7263.

[DSG15] G. S. Deaecto, M. Souza, and J. C. Geromel. “Discrete-Time Switched Linear Systems State Feedback Design With Application to Networked Control”. In: *IEEE Transactions on Automatic Control* 60.3 (Mar. 2015), pp. 877–881. DOI: 10.1109/TAC.2014.2341131.

[Dzu+16] D. Dzung, R. Guerraoui, D. Kozhaya, and Y.-A. Pignolet. “Never Say Never – Probabilistic and Temporal Failure Detectors”. In: *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*. May 2016, pp. 679–688. DOI: 10.1109/IPDPS.2016.92.

[EBH08] J. W. Eaton, D. Bateman, and S. Hauberg. *GNU Octave Manual*. 3rd. 2008.

[Fal+19a] J. Falk, F. Dürr, S. Linsenmayer, S. Wildhagen, B. Carabelli, and K. Rothermel. “Optimal Routing and Scheduling of Complemental Flows in Converged Networks”. In: *Proceedings of the 27th International Conference on Real-Time Networks and Systems*. RTNS ’19. Toulouse, France: ACM, 2019, pp. 154–164. DOI: 10.1145/3356401.3356415.

[Fal+19b] J. Falk, D. Hellmanns, B. W. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel. “NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++”. In: *Proceedings of the 2019 International Conference on Networked Systems*. NetSys. Mar. 2019, pp. 1–8. DOI: 10.1109/NetSys.2019.8854500.

[Fin+18] N. Finn, P. Thubert, B. Varga, and J. Farkas. *Deterministic Networking Architecture*. Internet-Draft draft-ietf-detnet-architecture-05. Work in Progress. Internet Engineering Task Force (IETF), May 2018. 41 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-detnet-architecture-05.

[FK15] Y. P. Fallah and M. K. Khandani. “Analysis of the Coupling of Communication Network and Safety Application in Cooperative Collision Warning Systems”. In: *Proceedings of the 6th ACM/IEEE International Conference on Cyber-Physical Systems*. ICCPS. Seattle, Washington: ACM, 2015, pp. 228–237. DOI: 10.1145/2735960.2735975.

[FLP85] M. J. Fischer, N. A. Lynch, and M. S. Paterson. “Impossibility of Distributed Consensus with One Faulty Process”. In: *Journal of the ACM* 32.2 (Apr. 1985), pp. 374–382. ISSN: 0004-5411. DOI: 10.1145/3149.214121.

[GAB19] A. Gujarati, M. Appel, and B. B. Brandenburg. “Achal: Building Highly Reliable Networked Control Systems”. In: *Proceedings of the International Conference on Embedded Software*. EMSOFT. New York, NY, USA: ACM, Oct. 2019. DOI: 10.1145/3349568.3351545.

[GCB08] J. C. Geromel, P. Colaneri, and P. Bolzern. "Dynamic Output Feedback Control of Switched Linear Systems". In: *IEEE Transactions on Automatic Control* 53.3 (Apr. 2008), pp. 720–733. DOI: 10.1109/TAC.2008.919860.

[GFB11] L. Greco, D. Fontanelli, and A. Bicchi. "Design and Stability Analysis for Anytime Control via Stochastic Scheduling". In: *IEEE Transactions on Automatic Control* 56.3 (Mar. 2011), pp. 571–585. ISSN: 0018-9286. DOI: 10.1109/TAC.2010.2058497.

[GH13] B. Galloway and G. P. Hancke. "Introduction to Industrial Control Networks". In: *IEEE Communication Surveys & Tutorials* 15.2 (2013), pp. 860–880. DOI: 10.1109/SURV.2012.071812.00124.

[GQD15] O. Gettings, S. Quinton, and R. I. Davis. "Mixed Criticality Systems with Weakly-hard Constraints". In: *Proceedings of the 23rd International Conference on Real Time and Networks Systems (RTNS'15)*. Lille, France: ACM, Nov. 2015, pp. 237–246. ISBN: 978-1-4503-3591-1. DOI: 10.1145/2834848.2834850.

[GRP16] K. Gatsis, A. Ribeiro, and G. J. Pappas. "Control-Aware Random Access Communication". In: *Proceedings of the 7th ACM / IEEE International Conference on Cyber-Physical Systems*. ICCPS. IEEE, Apr. 2016, pp. 1–9. DOI: 10.1109/ICCPS.2016.7479071.

[GSC08] E. Garone, B. Sinopoli, and A. Casavola. "LQG control over lossy TCP-like networks with probabilistic packet acknowledgements". In: *Proceedings of the 47th IEEE Conference on Decision and Control*. CDC. Cancun, Mexico, Dec. 2008, pp. 2686–2691. DOI: 10.1109/CDC.2008.4739460.

[Hee+10] W. P. M. H. Heemels, A. R. Teel, N. van de Wouw, and D. Nešić. "Networked control systems with communication constraints: Tradeoffs between transmission intervals, delays and performance". In: *IEEE Transactions on Automatic Control* 55.8 (Feb. 2010), pp. 1781–1796. DOI: 10.1109/TAC.2010.2042352.

[HM19] H. Howard and R. Mortier. "A Generalised Solution to Distributed Consensus". In: (Feb. 2019). arXiv: 1902.06776 [cs.DC].

[HNX07] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. "A Survey of Recent Results in Networked Control Systems". In: *Proceedings of the IEEE* 95.1 (Jan. 2007), pp. 138–162. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.887288.

[HRW15] J. Hwang, K. K. Ramakrishnan, and T. Wood. “NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms”. In: *IEEE Transactions on Network and Service Management* 12.1 (Mar. 2015), pp. 34–47. ISSN: 1932-4537. DOI: 10.1109/TNSM.2015.2401568.

[HW16] S. Hirche and K. Wehrle. *SPP 1914: Cyber-Physical Networking (CPN)*. Priority Programme – Deutsche Forschungsgemeinschaft (DFG). 2016. URL: https://gepris.dfg.de/gepris/projekt/273882191.

[IEE04] IEEE. *IEC/IEEE Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (Adoption of IEEE Std 1588-2008)*. Standard IEEE 1588 IEC 61588 First edition 2004-09. IEEE, Sept. 2004. 156 pp. DOI: 10.1109/IEEESTD.2004.95751.

[IEE16] IEEE. *IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks – Amendment 25: Enhancements for Scheduled Traffic*. IEEE Std 802.1Qbv-2015. IEEE, Mar. 2016. 57 pp. DOI: 10.1109/IEEESTD.2016.7440741.

[IYB06] O. C. Imer, S. Yüksel, and T. Başar. “Optimal control of LTI systems over unreliable communication links”. In: *Automatica* 42.9 (Sept. 2006), pp. 1429–1439. DOI: 10.1016/j.automatica.2006.03.011.

[Kha02] H. K. Khalil. *Nonlinear Systems*. 3rd ed. Pearson Education. Prentice Hall, 2002. ISBN: 9780130673893.

[KK12] K.-D. Kim and P. R. Kumar. “Cyber-Physical Systems: A Perspective at the Centennial”. In: *Proceedings of the IEEE* 100 (Mar. 2012). Special Issue, pp. 1287–1308. DOI: 10.1109/JPROC.2012.2189792.

[KPS02] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli. “Fault tolerance techniques for wireless ad hoc sensor networks”. In: *Proceedings of the IEEE International Conference on Sensors*. Vol. 2. IEEE, June 2002, pp. 1491–1496. DOI: 10.1109/ICSENS.2002.1037343.

[LA06] H. Lin and P. J. Antsaklis. “Switching Stabilization and l2 Gain Performance Controller Synthesis for Discrete-Time Switched Linear Systems”. In: *Proceedings of the 45th IEEE Conference on Decision and Control*. CDC. San Diego, CA, USA, Dec. 2006, pp. 2673–2678. DOI: 10.1109/CDC.2006.377641.

[LA09] H. Lin and P. J. Antsaklis. “Stability and Stabilizability of Switched Linear Systems: A Survey of Recent Results”. In: *IEEE Transactions on Automatic Control* 54.2 (Feb. 2009), pp. 308–322. ISSN: 0018-9286. DOI: 10.1109/TAC.2008.2012009.

[LA17] S. Linsenmayer and F. Allgöwer. "Stabilization of Networked Control Systems with Weakly Hard Real-Time Dropout Description". In: *Proceedings of the 56th IEEE Conference on Decision and Control*. CDC. Melbourne, Australia: IEEE, Dec. 2017, pp. 4765–4770. DOI: 10.1109/CDC.2017.8264364.

[LA18] S. Linsenmayer and F. Allgöwer. "Performance oriented triggering mechanisms with guaranteed traffic characterization for linear discrete-time systems". In: *Proceedings of the 2018 European Control Conference*. ECC. Limassol, Cyprus, 2018, pp. 1474–1479. DOI: 10.23919/ECC.2018.8550568.

[Lam98] L. Lamport. "The Part-time Parliament". In: *ACM Transactions on Computer Systems* 16.2 (May 1998), pp. 133–169. DOI: 10.1145/279227.279229.

[Lau18] B. Lauwens. *SimJulia.jl – A Discrete Event Process Oriented Simulation Framework Written In Julia (v0.7)*. 2018. URL: https://github.com/BenLauwens/SimJulia.jl.

[Lau79] A. J. Laub. "A Schur Method for Solving Algebraic Riccati Equations". In: *IEEE Transactions on Automatic Control* 24.6 (Dec. 1979), pp. 913–921. ISSN: 0018-9286. DOI: 10.1109/TAC.1979.1102178.

[LC15] Y. Li and M. Chen. "Software-Defined Network Function Virtualization: A Survey". In: *IEEE Access* 3 (2015), pp. 2542–2553. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2015.2499271.

[Lev+00] A. Levant, A. Pridor, R. Gitizadeh, I. Yaesh, and J. Z. Ben-Asher. "Aircraft Pitch Control via Second-Order Sliding Technique". In: *Journal of Guidance, Control, and Dynamics* 23.4 (2000), pp. 586–594. DOI: 10.2514/2.4591.

[LG04] X. Liu and A. Goldsmith. "Kalman Filtering with Partial Observation Losses". In: *Proceedings of the 43rd IEEE Conference on Decision and Control*. CDC. Nassau, Bahamas: IEEE, Dec. 2004, pp. 4180–4186. DOI: 10.1109/CDC.2004.1429408.

[Lin] Linux Foundation Project. *Data Plane Development Kit (DPDK)*. URL: http://dpdk.org.

[Lin+19] S. Linsenmayer, B. W. Carabelli, F. Dürr, J. Falk, F. Allgöwer, and K. Rothermel. "Integration of Communication Networks and Control Systems Using a Slotted Transmission Classification Model". In: *Proceedings of the 16th IEEE Annual Consumer Communications & Networking Conference*. CCNC. Jan. 2019, pp. 1–6. DOI: 10.1109/CCNC.2019.8651811.

[Lin+20] S. Linsenmayer, B. W. Carabelli, S. Wildhagen, K. Rothermel, and F. Allgöwer. “Controller and Triggering Mechanism Co-Design for Control over Time-Slotted Networks”. In: *IEEE Transactions on Control of Network Systems* (Sept. 2020). DOI: 10.1109/TCNS.2020.3024316.

[LSA71] A. H. Levis, R. A. Schlueter, and M. Athans. “On the behaviour of optimal linear sampled-data regulators”. In: *International Journal of Control* 13.2 (1971), pp. 343–361. DOI: 10.1080/00207177108931949.

[LSP82] L. Lamport, R. Shostak, and M. Pease. “The Byzantine generals problem”. In: *ACM Transactions on Programming Languages and Systems* 4.3 (1982), pp. 382–401.

[LWA16] K. D. Listmann, P. Wenzelburger, and F. Allgöwer. “Industrie 4.0 – (R)evolution ohne Regelungstechnik?” In: *at - Automatisierungstechnik* 64.7 (July 2016), pp. 507–520. DOI: 10.1515/auto-2016-0039.

[Mah+02] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. “Inferring Link Weights using End-to-End Measurements”. In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*. 2002, pp. 231–236. DOI: 10.1145/637201.637237.

[Mam+18] M. H. Mamduhi, J. S. Baras, K. H. Johansson, and S. Hirche. “State-Dependent Data Queuing in Shared-Resource Networked Control Systems”. In: *Proceedings of the 57th Conference on Decision and Control*. CDC. Miami, FL, USA: IEEE, Dec. 2018, pp. 1731–1737. DOI: 10.1109/CDC.2018.8619752.

[Mar+10] P. Martí, A. Camacho, M. Velasco, and M. E. M. B. Gaïd. “Runtime Allocation of Optional Control Jobs to a Set of CAN-Based Networked Control Systems”. In: *IEEE Transactions on Industrial Informatics* 6.4 (Nov. 2010), pp. 503–520. ISSN: 1551-3203. DOI: 10.1109/TII.2010.2072961.

[Mar+14] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. “ClickOS and the Art of Network Function Virtualization”. In: *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*. NSDI’14. Seattle, WA: USENIX Association, 2014, pp. 459–473. ISBN: 978-1-931971-09-6.

[McK+08] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. “OpenFlow: Enabling Innovation in Campus Networks”. In: *ACM SIGCOMM Computer Communication Review* 38.2 (Mar. 2008), pp. 69–74. ISSN: 0146-4833. DOI: 10.1145/1355734.1355746.

[MGS13] Y. Mo, E. Garone, and B. Sinopoli. "LQG Control with Markovian Packet Loss". In: *Proceedings of the European Control Conference*. ECC. Zurich, Switzerland, July 2013, pp. 2380–2385.

[MH09] A. Molin and S. Hirche. "On LQG joint optimal scheduling and control under communication constraints". In: *Proceedings of the 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*. CDC/CCC. Shanghai, P.R. China, Dec. 2009, pp. 5832–5838. DOI: 10.1109/CDC.2009.5400528.

[MH14] A. Molin and S. Hirche. "Price-Based Adaptive Scheduling in Multi-Loop Control Systems With Resource Constraints". In: *IEEE Transactions on Automatic Control* 59.12 (Dec. 2014), pp. 3282–3295. DOI: 10.1109/TAC.2014.2351892.

[MOS18] MOSEK ApS. *MOSEK Optimization Suite Release 8.1.0.64*. 2018. URL: https://docs.mosek.com/8.1/intro/index.html.

[NDR16] N. G. Nayak, F. Dürr, and K. Rothermel. "Time-Sensitive Software-Defined Network (TSSDN) for Real-Time Applications". In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France: ACM, 2016, pp. 193–202. ISBN: 978-1-4503-4787-7. DOI: 10.1145/2997465.2997487.

[NT04] D. Nešić and A. R. Teel. "Input-output stability properties of networked control systems". In: *IEEE Transactions on Automatic Control* 49.10 (Oct. 2004), pp. 1650–1667. ISSN: 0018-9286. DOI: 10.1109/TAC.2004.835360.

[NW09] Y. (Nie and X. Wu. "Shortest path problem considering on-time arrival probability". In: *Transportation Research Part B: Methodological* 43.6 (2009), pp. 597–613. ISSN: 0191-2615. DOI: 10.1016/j.trb.2009.01.008.

[OL88] B. M. Oki and B. H. Liskov. "Viewstamped Replication: A New Primary Copy Method to Support Highly-available Distributed Systems". In: *Proceedings of the 7th ACM Symposium on Principles of Distributed Computing*. 1988, pp. 8–17. DOI: 10.1145/62546.62549.

[ONF15] ONF. *OpenFlow Switch Specification Version 1.5.1*. Tech. rep. ONF TS-025. Open Networking Foundation, Mar. 2015.

[OO14] D. Ongaro and J. Ousterhout. "In Search of an Understandable Consensus Algorithm". In: *USENIX Annual Technical Conference*. Philadelphia, PA: USENIX Association, June 2014, pp. 305–319.

[Pap+03] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot. "Measurement and Analysis of Single-hop Delay on an IP Backbone Network". In: *IEEE Journal on Selected Areas in Communications* 21.6 (2003), pp. 908–921. DOI: 10.1109/JSAC.2003.814410.

[PG93] A. K. Parekh and R. G. Gallager. "A generalized processor sharing approach to flow control in integrated services networks: the single-node case". In: *IEEE/ACM Transactions on Networking* 1.3 (June 1993), pp. 344–357. DOI: 10.1109/90.234856.

[QN12] D. E. Quevedo and D. Nešić. "Robust stability of packetized predictive control of nonlinear systems with disturbances and Markovian packet losses". In: *Automatica* 48.8 (2012), pp. 1803–1811. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2012.05.046.

[QSG07] D. E. Quevedo, E. I. Silva, and G. C. Goodwin. "Packetized Predictive Control over Erasure Channels". In: *Proceedings of the American Control Conference*. ACC. 2007, pp. 1003–1008. DOI: 10.1109/ACC.2007.4282630.

[Ram+11] C. Ramesh, H. Sandberg, L. Bao, and K. H. Johansson. "On the dual effect in state-based scheduling of networked control systems". In: *Proceedings of the American Control Conference (ACC)*. San Francisco, CA, USA, June 2011, pp. 2216–2221. DOI: 10.1109/ACC.2011.5990925.

[RAZ16] A. Roy, H. Aydin, and D. Zhu. "On Task Period Assignment in Multiprocessor Real-Time Control Systems". In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France: ACM, 2016, pp. 151–160. ISBN: 978-1-4503-4787-7. DOI: 10.1145/2997465.2997469.

[RDG14] K. Rajawat, E. Dall'Anese, and G. B. Giannakis. "Dynamic Network Delay Cartography". In: *IEEE Transactions on Information Theory* 60.5 (May 2014), pp. 2910–2920. ISSN: 0018-9448. DOI: 10.1109/TIT.2014.2311802.

[Rei+12] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker. "Abstractions for Network Update". In: *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM '12. Helsinki, Finland, 2012, pp. 323–334. DOI: 10.1145/2342356.2342427.

[RSJ13] C. Ramesh, H. Sandberg, and K. H. Johansson. "Design of State-Based Schedulers for a Network of Control Loops". In: *IEEE Transactions on Automatic Control* 58.8 (Aug. 2013), pp. 1962–1975. ISSN: 0018-9286. DOI: 10.1109/TAC.2013.2251791.

[SA11] T. Samad and A. M. Annaswamy, eds. *The Impact of Control Technology*. IEEE Control Systems Society, Technical Report, 2011. URL: www.ieeecss.org.

[Saa+17] W. Saab, M. Mohiuddin, S. Bliudze, and J.-Y. Le Boudec. "Quarts: Quick Agreement for Real-time Control Systems". In: *Proceedings of the 22nd IEEE Conference on Emerging Technologies & Factory Automation*. ETFA. Sept. 2017, pp. 1–8. DOI: 10.1109/ETFA.2017.8247590.

[Sau+06] O. Saukh, P. J. Marrón, A. Lachenmann, M. Gauger, D. Minder, and K. Rothermel. "Generic Routing Metric and Policies for WSNs". In: *Proceedings of the 3rd European Workshop for WSNs (EWSN)*. Vol. 3868. LNCS. Feb. 2006, pp. 99–114. DOI: 10.1007/11669463_10.

[Sch+07] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. S. Sastry. "Foundations of Control and Estimation Over Lossy Networks". In: *Proceedings of the IEEE* 95.1 (Jan. 2007), pp. 163–187. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.887306.

[Sch+17] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl. "ILP-based Joint Routing and Scheduling for Time-Triggered Networks". In: *Proceedings of the 25th International Conference on Real-Time Networks and Systems*. RTNS '17. Grenoble, France, Oct. 2017. DOI: 10.1145/3139258.3139289.

[Sch08] L. Schenato. "Optimal Estimation in Networked Control Systems Subject to Random Delay and Packet Drop". In: *IEEE Transactions on Automatic Control* 53.5 (Aug. 2008), pp. 1311–1317. DOI: 10.1109/TAC.2008.921012.

[Sch09] L. Schenato. "To Zero or to Hold Control Inputs with Lossy Links?" In: *IEEE Transactions on Automatic Control* 54.5 (2009), pp. 1093–1099. DOI: 10.1109/TAC.2008.2010999.

[Sin+04] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. "Kalman Filtering With Intermittent Observations". In: *IEEE Transactions on Automatic Control* 49.9 (Sept. 2004), pp. 1453–1464. DOI: 10.1109/TAC.2004.834121.

[Sin+05] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S. S. Sastry. "Optimal control with unreliable communication: the TCP case". In: *Proceedings of the American Control Conference*. ACC. IEEE. 2005, pp. 3354–3359. DOI: 10.1109/ACC.2005.1470488.

[Sin+08] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, and S. S. Sastry. "Optimal linear LQG control over lossy networks without packet acknowledgment". In: *Asian Journal of Control* 10.1 (2008), pp. 3–13. DOI: 10.1002/asjc.1.

[Siv+13] A. Sivaraman, K. Winstein, S. Subramanian, and H. Balakrishnan. "No Silver Bullet: Extending SDN to the Data Plane". In: *Proceedings of the 12th ACM Workshop on Hot Topics in Networks (HotNets-XII)*. College Park, Maryland, Nov. 2013, 19:1–19:7. DOI: 10.1145/2535771.2535796.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. *Specification of Guaranteed Quality of Service*. RFC 2212. IETF, Sept. 1997. URL: http://tools.ietf.org/rfc/rfc2212.txt.

[SSP15] S. K. Singh, R. Singh, and B. C. Pal. "Stability Analysis of Networked Control in Smart Grids". In: *IEEE Transactions on Smart Grid* 6.1 (2015), pp. 381–390. DOI: 10.1109/tsg.2014.2314494.

[ST17] M. van Steen and A. S. Tanenbaum. *Distributed Systems*. 3rd. 2017. ISBN: 978-15-430573-8-6.

[SWK09] U. Schmid, B. Weiss, and I. Keidar. "Impossibility Results and Lower Bounds for Consensus under Link Failures". In: *SIAM Journal of Computing* 38.5 (2009), pp. 1912–1951. DOI: 10.1137/S009753970443999X.

[VH08] A. Varga and R. Hornig. "An overview of the OMNeT++ simulation environment". In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques*. SIMUTools. 2008, 60:1–60:10.

[WM03] M. Welzl and M. Mühlhäuser. "Scalability and quality of service: a trade-off?" In: *IEEE Communications Magazine* 41.6 (June 2003), pp. 32–36. DOI: 10.1109/MCOM.2003.1204745.

[WYB02] G. C. Walsh, H. Ye, and L. G. Bushnell. "Stability analysis of networked control systems". In: *IEEE Transactions on Control Systems Technology* 10.3 (May 2002), pp. 438–446. DOI: 10.1109/87.998034.

[Xu+14] Y. Xu, K.-E. Årzén, E. Bini, and A. Cervin. "Response Time Driven Design of Control Systems". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 6098–6104. ISSN: 1474-6670. DOI: 10.3182/20140824-6-ZA-1003.00289.

[Yu+13] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha. "FlowSense: Monitoring Network Utilization with Zero Measurement Cost". In: *Proceedings of the 14th International Conference on Passive and Active Measurement*. Ed. by M. Roughan and R. Chang. Berlin, Heidelberg: Springer, Mar. 2013, pp. 31–41. DOI: 10.1007/978-3-642-36516-4_4.

[Zha+08] F. Zhang, K. Szwaykowska, W. Wolf, and V. Mooney. "Task Scheduling for Control Oriented Requirements for Cyber-Physical Systems". In: *Proceedings of the 2008 Real-Time Systems Symposium*. RTSS. IEEE, Nov. 2008, pp. 47–56. DOI: 10.1109/RTSS.2008.52.

[Zha+19] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng. "Networked control systems: a survey of trends and techniques". In: *IEEE/CAA Journal of Automatica Sinica* 7.1 (July 2019), pp. 1–17. DOI: 10.1109/JAS.2019.1911651.

[Zie01] M. Ziegelmann. "Constrained shortest paths and related problems". PhD thesis. Saarländische Universitäts- und Landesbibliothek, 2001.

[Zin16] S. Zinkler. "In-network Packet Priority Adaptation for Networked Control Systems". Code available at https://github.com/znsn/masterthesis. Master's thesis. University of Stuttgart, Aug. 2016.

[ZJ08] Y. Zhang and J. Jiang. "Bibliographical review on reconfigurable fault-tolerant control systems". In: *Annual Reviews in Control* 32.2 (2008), pp. 229–252. ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2008.03.008.

[ZY10] W.-A. Zhang and L. Yu. "Stabilization of Sampled-Data Control Systems With Control Inputs Missing". In: *IEEE Transactions on Automatic Control* 55.2 (Feb. 2010), pp. 447–452. DOI: 10.1109/TAC.2009.2036325.

Erklärung

Ich erkläre hiermit, dass ich, abgesehen von den ausdrücklich bezeichneten Hilfsmitteln und den Ratschlägen von jeweils namentlich aufgeführten Personen, die Dissertation selbstständig verfasst habe.

(Ben William Carabelli)