Model-Predictive Control in Communication Networks

Dissertation Submitted to the Department of Mathematics and Computer Science at the Freie Universität Berlin in Partial Fulfillment of the Requirements for the Degree *Doctor rerum naturalium* (Dr. rer. nat.)

Richard Schöffauer

Berlin 2023

Primary Reviewer and Supervisor:

Prof. Dr.-Ing. habil. Gerhard Wunder

External Reviewer:

Prof. Dr.-Ing. James Gross

Date of Disputation:

13.10.2023

Statement of Originality:

I herewith declare that I wrote and composed this dissertation, titled "Model-Predictive Control in Communication Networks", independently. I did not use any other sources than the ones stated in the bibliography. Furthermore I declare that this work has never before been submitted by me or somebody else at this or any other university.

(Date & Signature, Dipl.-Ing. Dipl.-Ing. Richard Schöffauer)

Statement of Contribution:

I herewith declare that all parts of this dissertation, titled "Model-Predictive Control in Communication Networks", specifically including the papers of which it consists, have been written solely and are the result of research done solely by the author, Dipl.-Ing. Dipl.-Ing. Richard Schöffauer, if not clearly marked otherwise with a green line on the left side of the page.

(Date & Signature, Prof. Dr.-Ing. habil. Gerhard Wunder)

Contents

Introduction

This dissertation is of cumulative form and therefore primarily composed of the contents of 8 papers (see Section 0.1), all of which are the result of work conducted under the research program "Cyber-Physical Networking", financed through the DFG in priority program SPP 1914. As indicated by its name, the now completed program addressed open questions concerning control of and communication inside networks, especially when the two are intertwined. The goal of the program can be described as follows: Developing new analytic frameworks in which plant control and communication (either between multiple controllers or within the control loop) are designed jointly. In contrast, a conventional design approach would see both issues being handled separately.

The program consisted of several research groups with individual projects. The project responsible for the development of this dissertation is labeled "Model-Predictive Cyber-Physical Networking" and was worked on by

- Dipl.-Ing. Dipl.-Ing. Richard Schöffauer (author of this dissertation) and
- Prof. Dr.-Ing. habil. Gerhard Wunder (supervisor of this dissertation)

from the Freie Universität Berlin, and a collaborating research group, consisting of

- Dipl.-Ing. Jannik Hahn and
- Prof. Dr.-Ing. Olaf Stursberg

from the Universität Kassel. While the Freie Universität Berlin would field expertise in the communication domain, the Universität Kassel would do so for the control domain.

The core idea behind the "Model-Predictive Cyber-Physical Networking" project revolves around predicting communication delay and letting the plant controllers use this prediction to improve their performance. In its simplest form, the set-up consists of a set of plant controllers (that are coupled to a certain degree) and a communication network over which the controllers exchange information. As a basic example, one might think of the control of a car platoon where each car (plant) has its separate dynamics but the control decisions are coupled through a constraint involving the minimum and maximum distance between two neighboring cars in order to maintain formation and avoid collisions. Conventionally, one would design a communication solution that guarantees a certain worst-case communication delay with which data is exchanged between the cars and then, based on this worst-case delay, design the plant controllers accordingly. In the project's joint design approach, however, we assume that the communication delay is predictable to some degree and design a communication solution that exploits this fact to not only minimize the worst-case delay but to predict the immediate delay over the next few time-steps. These delay forecasts are then communicated to the cars (plants) which make use of them and improve their control decisions, thereby improving the overall control performance. Therefore, the research focuses on two problems:

1. How can the delay be predicted?

2. And how can the delay forecasts be utilized by the controllers?

The papers contained in this dissertation are focused on the first question and thus on the communication domain. The second question was investigated by our research partners from the Universität Kassel and therefore does not lie in the scope of this dissertation. However, there are 2 papers that are collaborative works and consequently also investigate problems from the control domain. Since their content is very hard to grasp for readers that are not familiar with the topic, we have dedicated Section 0.4 to the explanation of the main mechanisms through which the plant controllers may utilize the delay forecasts.

0.1 List of Papers

This dissertation consists of 8 papers out of which

- 2 Papers have been published in peer-reviewed journals,
- 1 Paper has been submitted for publication in a peer-reviewed journal,
- 4 Papers have been published in peer-reviewed conferences,
- and 1 Paper has been published on an open-access website.

The journals and conferences fall under the umbrella of the IEEE (Institute of Electrical and Electronics Engineers) and the IFAC (International Federation of Automatic Control) which are the largest technical professional organizations in their respective fields.

All Papers are co-authored, 6 of them solely by Prof. Dr.-Ing. habil. Gerhard Wunder who is the supervisor of this dissertation. 2 papers are also co-authored by J. Hahn and O. Stursberg and do contain parts without any contribution from the dissertation's author. These parts are clearly marked with a green line on the side and make up roughly 10% of the overall content of this dissertation.

Rather than using chronological order, we will present the papers based on their content, facilitating the following separation into 3 categories:

Predictive Control for Queueing Networks

Paper 1. R. Schoeffauer, G. Wunder
"Predictive Network Control and Throughput Sub-Optimality of MaxWeight" (Predictive Network Control)
IEEE European Conference on Networks and Communications (EuCNC), 2018

Paper 2. R. Schoeffauer, G. Wunder
"Model-Predictive Control for Discrete-Time Queueing Networks With Varying Topology" (Throughput Optimality of PNC)
IEEE Transactions on Control of Network Systems (TCNS), 2021

Paper 3. R. Schoeffauer, G. Wunder
"Stability Results on Synchronized Queues in Discrete-Time for Arbitrary Dimension" (Assembly-Queues)
arXiv, 2020

Reliable Predictive Algorithms

Paper 4. J. Hahn, R. Schoeffauer, G. Wunder, O. Stursberg
"Distributed MPC with Prediction of Time-Varying Communication Delay" (Quadratic Reliable Prediction)
IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys), 2018

Paper 5. R. Schoeffauer, G. Wunder
"A Linear Algorithm for Reliable Predictive Network Control" (Linear Reliable Prediction)
IEEE Global Communications Conference (GLOBECOM), 2018

Predictive Control for Age-of-Information

Paper 6. J. Hahn, R. Schoeffauer, G. Wunder and O. Stursberg
"Using AoI Forecasts in Communicating and Robust Distributed Model-Predictive Control" (Prediction of AoI)
IEEE Transactions on Control of Network Systems (TCNS), 2021

Paper 7. R. Schoeffauer, G. Wunder
"An Algorithm for Exact Numerical Age-of-Information Evaluation in Multi-Agent Systems" (State-Space of AoI)
IEEE International Conference on Communications (ICC), 2022

Paper 8. R. Schoeffauer, G. Wunder
"Age-of-Information in Clocked Networks" (AoI in Clocked Networks)
In Submission, IEEE/ACM Transactions on Networking

0.2 Copyright Agreements

In accordance with the IEEE Copyright, we have to make the following notice for the corresponding papers:

Furthermore, we refer to the following message by the publisher:

0.3 Structure & Notation

The structure of this dissertation follows a simple pattern. Ignoring first and last chapters, each chapter contains one of the papers from the list in Section 0.1 as its main content together with preliminary and concluding remarks. These remarks serve as a guide that leads the reader from one paper to the next and shed some light on selective additional context necessary to understand the paper since each paper was written with a certain audience in mind (which usually was already familiar with concepts referred to in the paper). Additionally, they also critically discuss weak points in the paper and may extend on certain concepts that were cut short (usually due to page limitation).

The notation in the original papers is not harmonized. While the earlier papers are written from the perspective of a novice researcher, later papers profit from the author having read a significant amount of papers of similar content. For this dissertation, the author tried to harmonize the notation in the chapters as much as possible since this drastically improves accessibility of the content. However, as this directly violates the doctorate rules and regulations set by the department and the university, the papers are once again printed in the appendix, this time in their original version.

As the title of this dissertation suggests, prediction of network states is a fundamental method applied throughout the presented papers. If we let q_t denote a state-vector at time-step t, then a prediction of that state made in time-step $t_0 < t$ should be denoted as

$$q_{t|t_0} \tag{1}$$

Indeed, this notation is used in the field of Model-Predictive Control and therefore can be found in Papers 4 and 6 (where it is used in those parts that contain no contribution from the dissertation's author). Note that $q_{t|t_0}$ is not to be confused with $\mathbb{E}[q_t|I_{t_0}]$ (where I_{t_0} stands for whatever information is available in time-step t_0), as the prediction may be performed over a totally different system-model, e.g. in order to reduce computation time. In this dissertation, we will use a slightly different notation for two reasons:

1. In Paper 2 we utilize a system-model together with *two* different prediction-models, already forcing us to find an additional way to distinguish between $q_{t|t_0}$ from one prediction-model and $q_{t|t_0}$ from another one.

2. As we only treat time-invariant systems and their control, w.l.o.g. we can always assume that the current time-step, i.e. the time-step in which the prediction of future states is performed, is $t_0 = 0$. However, always entailing the zero in $q_{t|0}$ just makes the formulas harder to read without revealing any important information.

Therefore, we will write $\dot{q}_t$ to denote a prediction instead of $q_{t|t_0}$. This notation implicitly assumes that $t_0 = 0$ and stresses the fact that $\dot{q}_t$ could result from an entirely different system-model (which we usually refer to as the prediction-model). We do not make use of any time derivatives, so there is no opportunity for the reader to confuse $\dot{q}_t$ with $\frac{dq}{dt}$.

As a final remark on the notation, note that we will typically use lower indices (x_t) to denote the time index. An upper index is used to count elements of a set

($\mathcal{X} = \{x^1, x^2, x^3\}$) or vector ($x = (x^1, x^2, x^3)^\intercal$). Whenever the upper index is indeed an exponent, it is usually very clear from the context and mentioned in the text immediately before or after the occurrence.

0.4 Delay Forecasts in the Control Domain

As already mentioned, this section tries to cover the most important aspects of the control realm. This is necessary to justify the dissertation's focus on network prediction and complete the scope of the overarching research project.

To facilitate exploitation of delay forecasts in the control domain, we employ model predictive controllers (MPCs). What differentiates MPCs from conventional controllers is the fact that they explicitly evaluate the impact of the control decision on the future system behavior. Let the equation

$$x_{t+1} = Ax_t + Bu_t + Dd_t \tag{2}$$

denote a linear, discrete-time system evolution with x_t, u_t, d_t being state-, control-, disturbance-vector of arbitrary dimension and A, B, D being matrices that project the corresponding vectors into the state-space. For simplicity, we assume that A, B, D are invertible. Like all controllers, an MPC takes the currently available information (state x_t and possibly all quantities from the past) and produces a control-vector u_t. It does so by optimizing a predefined objective (e.g. a cost function) that is a function not only of the immeditate next state x_{t+1} but also of future states $x_{t+2}, x_{t+3}, \ldots x_{t+H}$. Of course, these future states are unknown in time-step t and therefore the MPC controller uses a prediction-model to obtain an estimate (or prediction) of these states. This is why H is called the prediction horizon. Naturally, it is prudent to use a prediction-model that is very similar to the actual system evolution (2). In our example, a prediction of future states can be developed by iteratively applying (2) which yields for $t = 0$ and $H = 3$:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} Ax_0 \\ A^2x_0 \\ A^3x_0 \end{pmatrix} + \begin{pmatrix} B & & \\ AB & B & \\ A^2B & AB & B \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix} + \begin{pmatrix} D & & \\ AD & D & \\ A^2D & AD & D \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ d_2 \end{pmatrix} \tag{3}$$

(Throughout the dissertation, we will leave zero-entries in matrices vacant as long as it supports readability.)

The disturbances d_0, d_1 and d_2 are unknown at time-step $t = 0$. However, as time progresses these quantities can be observed. E.g. in time-step $t = 2$, disturbance d_0 can be deduced from the past observation as $d_0 = D^{-1}(x_1 - Ax_0 - Bu_0)$. Hence, when applying u_1 we will have knowledge of d_0 and thus u_1 can be a function of d_0. Though this dependency could take any shape, it is prudent to make it linear in order for the optimization of the cost function to remain practical. Hence we can define the dependency as

$$\begin{pmatrix} \dot{u}_0 \\ \dot{u}_1 \\ \dot{u}_2 \end{pmatrix} = \begin{pmatrix} 0 & & \\ F_{1,0} & 0 & \\ F_{2,0} & F_{2,1} & 0 \end{pmatrix} \begin{pmatrix} \dot{d}_0 \\ \dot{d}_1 \\ \dot{d}_2 \end{pmatrix} + \begin{pmatrix} \dot{v}_0 \\ \dot{v}_1 \\ \dot{v}_2 \end{pmatrix} \tag{4}$$

where quantities are now denoted with a dot $\dot{}$ on top because they are no longer part of the actual system evolution. (There is no connection to the frequently used notation for time derivatives.) They are part of the prediction-model, since splitting the control u into a term for linear disturbance feedback $F_{i,j} \cdot \dot{d}$ and an offset $\dot{v}$ is a simplification that does not have to hold true for the actual system evolution in the end. It is simply a way for us to efficiently obtain a prediction of future system-states. We also do not know the exact disturbances d that will impact the system in future time steps. Hence, they are, for now, replaced by $\dot{d}$. Plugging this substitution into (3) yields

$$\begin{aligned}\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix} &= \begin{pmatrix}Ax_0\\ A^2x_0\\ A^3x_0\end{pmatrix} + \begin{pmatrix}B & & \\ AB & B & \\ A^2B & AB & B\end{pmatrix}\begin{pmatrix}\dot{v}_0\\ \dot{v}_1\\ \dot{v}_2\end{pmatrix} \\ &+ \left[\begin{pmatrix}B & & \\ AB & B & \\ A^2B & AB & B\end{pmatrix}\begin{pmatrix}0 & & \\ F_{1,0} & 0 & \\ F_{2,0} & F_{2,1} & 0\end{pmatrix} + \begin{pmatrix}D & & \\ AD & D & \\ A^2D & AD & D\end{pmatrix}\right]\begin{pmatrix}\dot{d}_0\\ \dot{d}_1\\ \dot{d}_2\end{pmatrix}\end{aligned} \tag{5}$$

Again, we have to replace x with $\dot{x}$ because the states on the LHS are not necessarily the states that will occur in the future, they are just a suitable prediction of them. Now that the controller has obtained a prediction of future system-states, it can use them to optimize an objective that typically revolves around minimizing the distance between the states and a reference r_t. A common control objective could look like this:

$$\min \left[\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix} - \begin{pmatrix}r_1\\ r_2\\ r_3\end{pmatrix}\right]^{\intercal} Q \left[\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix} - \begin{pmatrix}r_1\\ r_2\\ r_3\end{pmatrix}\right] \tag{6}$$

with some semi-definite matrix Q.

The minimization is performed over the offsets $\dot{v}$ but also over the feedback matrices $F_{i,j}$ since they were arbitrarily introduced by us. The disturbances $\dot{d}$ are usually replaced by their bounds such that the optimization yields values for $\dot{v}$ and $F_{i,j}$ which abide to given constraints even in the worst case scenarios.

To recap, the MPC optimizes this objective, using the prediction-model, and obtains the vectors $\dot{v}_t$ and matrices $F_{i,j}$. Both can be combined to yield a control trajectory consisting of $\dot{u}_0, \dot{u}_1, \dot{u}_2$. Next, the controller sets $u_0 = \dot{u}_0$, i.e. applies the first control of the control trajectory to the actual system (2). However, it does not use $\dot{u}_1$ or $\dot{u}_2$ for the upcoming time-steps but instead repeats the entire process, resulting in an implicit control law.

Having established the basic principle via which an MPC operates, we can get to the real question at hand: how can the MPC controller use delay forecasts to improve its control decision. For that, we need to look back at the feedback matrix for the disturbances. Assuming that $H = 5$ we get

$$\begin{pmatrix}\dot{u}_0\\ \dot{u}_1\\ \dot{u}_2\\ \dot{u}_3\\ \dot{u}_4\end{pmatrix} = \underbrace{\begin{pmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \blacksquare & \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0}\\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \mathbf{0}\end{pmatrix}}_{\text{Feedback Matrix}} \begin{pmatrix}\dot{d}_0\\ \dot{d}_1\\ \dot{d}_2\\ \dot{d}_3\\ \dot{d}_4\end{pmatrix} + \begin{pmatrix}\dot{v}_0\\ \dot{v}_1\\ \dot{v}_2\\ \dot{v}_3\\ \dot{v}_4\end{pmatrix} \tag{7}$$

For simplicity, we substituted the matrices $F_{i,j}$ with black squares. Notice how $\dot{u}_0$ cannot make use of any disturbance feedback, since a disturbance in time-step t can *at best* be observed in time-step $t+1$. This is why, in the feedback matrix, elements on the diagonal and above are zero matrices. However, the illustrated population of the matrix is the best-case scenario. It might take even longer to determine a past disturbance, e.g. when the last observed system-state is also several time-steps old. For our scenario, the disturbances can be interpreted as system-states from other controllers that are communicated over a network with delay. If the worst-case delay grows, then feedback is delayed and entries from the feedback matrix turn to zero. e.g. if the worst case delay is 3, then $\dot{u}_3$ is the first control that has knowledge of $\dot{d}_0$, as illustrated in the following examples:

$$\begin{array}{ccc}
\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \mathbf{0} \end{pmatrix} &
\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0} \end{pmatrix} &
\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \\
\text{Worst-Case Delay} = 1 & \text{Worst-Case Delay} = 2 & \text{Worst-Case Delay} = 3
\end{array}$$

However, using delay forecasts does allow us to fracture this structure, as now every time-step is given its separate delay value (which logically is as good or better as the worst-case delay). This facilitates the repopulation of the feedback matrix to some degree as the following comparison illustrates:

$$\begin{array}{cc}
\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} &
\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \mathbf{0} \end{pmatrix} \\
\text{Delay} = \{3,3,3,3,3\} & \text{Delay} = \{2,3,2,1,3\}
\end{array}$$

This time, the delay is given via a trajectory such that $\{2,3,2,1,3\}$ expresses that the delay is 2 in time-step $t=0$, 3 in time-step $t=1$, 2 in time-step 2, and so on. This way, the MPC has more freedom, i.e. more variables, over which to optimize the objective and can thus potentially obtain an improved control-vector (compared to the MPC controller that can only count on the worst-case delay).

This completes the first chapter of this dissertation. A separate treatment of the project's main concepts of the network control aspect is not necessary since each of the following papers is self-contained and therefore already includes an introduction.

Paper 1

Predictive Network Control

Full Paper Title Predictive Network Control and Throughput Sub-Optimality of MaxWeight

Publisher IEEE European Conference on Networks and Communications (Eu-CNC), 2018

Authors Richard Schöffauer, Gerhard Wunder

Acknowledgment This paper is part of and thereby supported by the *priority program SPP1914 Cyber-Physical Networking* receiving funds from the DFG as well as the *Horizon 2020 project ONE5G* (ICT-760809) receiving funds from the European Union. The authors would like to acknowledge the contributions of their colleagues in the project, although the views expressed in this contribution are those of the authors and do not necessarily represent the project.

Abstract We present a novel control policy, called Predictive Network Control (PNC) to control wireless communication networks on packet level, based on paradigms of Model-Predictive Control (MPC). In contrast to common myopic policies, who use one step ahead prediction, PNC predicts the future behavior of the system for an extended horizon, thus facilitating performance gains. We define an advanced system-model in which we use a Markov chain in combination with a Bernoulli trial to model the stochastic components of the network. Furthermore, we introduce the algorithm and present two detailed simulation examples, which show general improved performance and a gain in stability region compared to the standard MaxWeight policy.

1.1 Preliminary Remarks

As the first published work, this paper's topic is entirely based upon the fundamental questions raised in the project proposal and therefore takes a first approach to bring prediction to the control of queueing networks. In order to do so, the paper proposes a new control policy and compares it to the well-known MaxWeight policy (MW). While it includes a sufficient introduction to queueing networks, it lacks any further details on the MW policy due to a page restriction and the target audience at the conference. However, since it is very much related to the proposed policy and arises various times in the other papers as well, it is prudent to give a fairly detailed description of MW at this point.

The MW policy operates on queueing networks, a detailed description of which is already given in the paper. Very concisely, a queueing network is expressed by the following system equation:

$$q_{t+1} = q_t + RM_t v_t + a_t \qquad \begin{aligned} q_t, a_t &\in \mathbb{N}_+^n \\ v_t &\in \{0,1\}^m \\ R_t &\in \{-1,0,1\}^{n\times m} \end{aligned} \tag{1.1}$$

The system-state q_t is subject to an influx term, consisting of the so-called arrival a_t and an efflux term $RM_t v_t$. The matrix product RM_t maps a binary control-vector v_t into the state-space. While R is fixed, M_t is a temporal realization of a probabilistic counterpart. Furthermore, v_t is constrained by $Cv_t \leq c$. The goal is to prevent the system-state q_t from growing to infinity.

Usually, M_t is independent of q_0 and is the result of a Bernoulli trial (component-wise) on a matrix $\bar{M}$ such that $\bar{M} = \mathbb{E}[M_t]$. The MW policy is based on the simple quadratic minimization problem

$$\begin{aligned} &\min_{v_0} \ \mathbb{E}[q_1^\intercal q_1 \,|\, q_0] \\ =& \min_{v_0} (q_0 + a_0)^\intercal (q_0 + a_0) + 2\,(q_0 + a_0)^\intercal R\bar{M}v_0 + v_0^\intercal \bar{M}^\intercal R^\intercal R\bar{M}v_0 \\ &\quad \text{s.t.} \qquad Cv_0 \leq c, \quad q_1 \geq 0 \end{aligned} \tag{1.2}$$

that tries to minimize the average squared queue-state of the next time-step (for simplicity, the current time-step is set to $t = 0$). Crucially, the quadratic term $v_0^\intercal \bar{M}^\intercal R^\intercal R\bar{M}v_0$ in (1.2) is bounded because the values of all quantities cannot exceed 1 per definition. In contrast, the linear term $2\,(q_0 + a_0)^\intercal R\bar{M}v_0$ from the same equation can grow to infinity due to the queue-state q_0, which is not bounded. Hence, given large enough q_0, the optimal control v_0^* for solving (1.2) is almost certainly the control that minimizes the linear term by itself. The MW policy (interpreted as a function $\phi(q_0)$) is therefore defined as

$$\text{MW:} \qquad \phi^{\text{MW}}(q_0) := \arg\min_{v_0} \ q_0^\intercal R\bar{M}v_0 \qquad \text{s.t.} \qquad \begin{pmatrix} Cv_0 \leq c \\ q_0 + R^- v_0 \geq 0 \end{pmatrix} \tag{1.3}$$

where R^- is the routing matrix R without any positive entries so that $q_1 \geq 0$ can be guaranteed no matter the realization of M_t.

It is prudent to define the *control-option* $u_t = R\bar{M}v_t$ to simplify the formulation. The control-option u_t can be interpreted as the expected impact of v_t on the state-space: $\mathbb{E}[q_1 \,|\, q_0] = q_0 + u_0 - a_0$, and thus represents the expected efflux. Using this notation, MW becomes

$$\phi^{\mathrm{MW}}(q_0) = \arg\min_{u_0 \in \mathcal{U}} q_0^{\intercal} u_0 \tag{1.4}$$

where $\mathcal{U} = \{u^0, u^1, \dots\}$ represents the set of all available control-options which is dictated by the constraints.

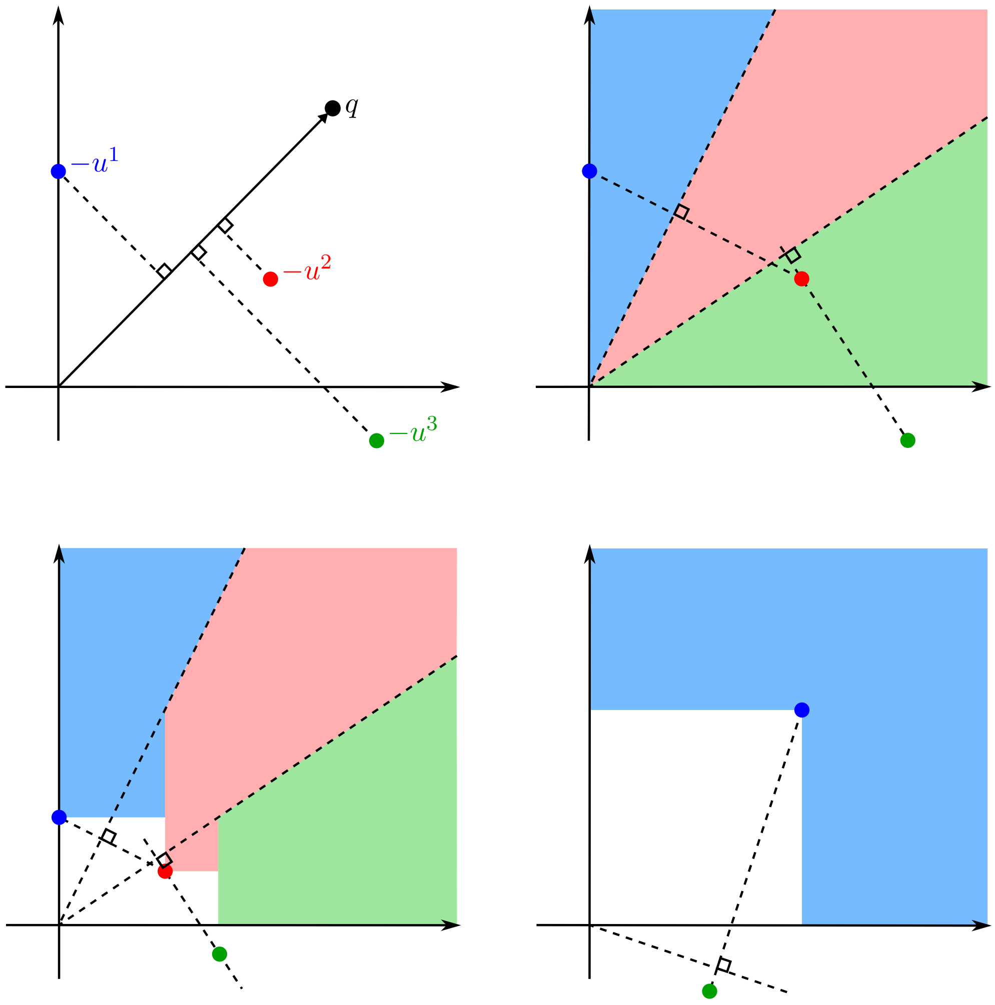


Fig. 1.1: Visualization of the MW policy. In a 2-dimensional state-space, MW chooses the control-option, whose negative vector has the largest projection on the state-vector.

It can be useful to visualize the MW policy through Figure 1.1. To simplify the matter as much as possible, we drop the time index. According to (1.4), MW chooses the very u whose negative vector yields the largest projection onto the state-vector q. Figure 1.1, top-left, depicts the state-space with a state-vector q inside. Obviously, out of the three given control-options, $-u^2$ produces the largest projection onto q, and thus MW maps this specific system-state q onto the control-option u^2. Determining the largest projection for every queue-state results in a separation of the state-space into areas, which map the state-vector q onto a control-option u as shown in the top-right where each colored area maps to an equally colored control-option.

Of course, we have to keep in mind the constraint $q_0 + R^{-}v_0 \geq 0$ which forbids certain control actions to be considered if q is too small, leading to a more complicated mapping when q is close to the origin of the state-space. This is illustrated in the bottom-left picture. (Compared to the previous pictures, now the zoom factor is changed, solely for the sake of improved visualization.) The bottom right picture depicts a case with a different set of control-options. This time, the green control-option is dominated by the blue one, i.e. there is no queue-state q for which the MW policy would activate the green control-option. (Queue-states cannot become negative.)

Finally, we have to mention that MW features throughput optimality as its most important property. For a succinct explanation, notice that the efflux in (1.1), i.e. the term RM_tv_t, is bounded. Hence, not every rate of influx, given by $\mathbb{E}[a_t]$, can be compensated by the efflux. E.g., if RM_t is such that the network only allows for one packet per time-step to leave the system (even under the best possible choices of v_t), an average arrival of two packets per time-step can never be compensated, no matter the policy. Consequently, the queues in the network would grow to infinity. A policy is called throughput optimal if it can stabilize the network (i.e. prevent the state from growing to infinity) for all arrival rates, which can be compensated by the network topology (given through $R\bar{M}$) at all.

1.2 Paper Body

1.2.1 Introduction and Motivation

Modern wireless networks, such as 5G, have an increasing amount of options to route packets to multiple nodes, making information flow control essential for throughput performance, e.g. to avoid bottlenecks due to cell overload, or to exploit diversity of wireless links for low latency communication [1]. Seminal work on wireless network control was published in [2] in 1992 and introduced the so-called MaxWeight policy (MW). This policy still stands as a benchmark and was improved upon many times, e.g. in [3] and [4]. However, MW and all its deviations are of the *myopic* type [5]. This means, they only make decisions based on immediate *next step* system changes (while using a time-discrete network description which is the most convenient model due to clocked devices in reality). Although not without its flaws (e.g. large delays of single packets under low workload [6]), this approach used to be reasonable due to limited amount of computational resources and high pace requirements. Especially in simple network topologies (e.g. broadcasting), these policies are well studied [7]. Nowadays though, with advancements in computational power, we have the option of using more mathematically ambitious algorithms to devise policies that can improve on the network behavior. Initial attempts in this direction were made in [8], where in each time-step a draining problem is solved to minimize delay for an OFDM broadcast channel.

In this paper, we introduce a new network control policy, that we call Predictive Network Control (PNC). Herein we use paradigms from the field of Model-Predictive Control (MPC) [9], to devise a policy that predicts the system behavior for multiple steps into the future. While we gain improvements in performance, this approach additionally produces a schedule of predicted communications which seems very intriguing for implementation into Cyber-Physical Systems, where control and communication merge together.

This work is dedicated to present the new policy together with the utilized system-model. We omit analytical results for later publications but provide numerical results which indicate inferiority of MW, in terms of performance and overall stability.

1.2.2 System Model

We concentrate on the fairly common case of a discrete-time, packet-level communication network, where we have an arbitrary number of entities sending information to one another [5]. Additionally, information may enter or leave the whole system. If an entity receives more information than it can send, it has to store it in a buffer. The amount of information in that buffer we call the queue length $q_t^{(i)}$ (buffer i, time-step t).

Considering all entities at once, we get the queue-vector $q_t \in \mathbb{N}^n$, n being the number of all buffers. It is possible that one entity keeps multiple buffers, e.g. if the received information is of different type. This will simply result in n being larger than the number of entities. Between the entities there exist so-called communication *links* as column vectors. E.g. the vector $\begin{pmatrix}-1 & 1 & 0 & \dots & 0\end{pmatrix}^\intercal$ would send information from the first buffer (decreasing its size by 1) to the second buffer (increasing its size by 1). These m vectors collected side by side form the routing-matrix R. We can influence the network by deciding, which links are to be activated at the current time. For that, we use the binary input- or control-vector $v_t \in \{0,1\}^m$. The system evolution can then be expressed as

$$q_{t+1} = q_t + RM_tv_t + a_t \tag{1.5}$$

where $R \in \mathbb{Z}^{n\times m}$ is discrete valued and $M_t \in \{0,1\}^{n_v\times n_v}$ is a diagonal matrix which is a temporal realization of a probabilistic counterpart. Information leaves the system, when columns are activated, that possess more negative than positive entries. This represents the information reaching its intended destination, which is the goal of any controller. In contrast, the arrival-vector $a_t \in \mathbb{N}^n$ represents an information influx by simply increasing q_t (usually in a stochastic way). Its time average $\bar{a} = \mathbb{E}[a] \in \mathbb{R}^n_+$ we call the arrival-rate, which is pivotal for system stability.

If every possible communication link (column of R) could be activated at the same time, there would be no need to steer the system. However, in reality some of these *links* might not be available at the same time, because they share the same communication *channels* (which have a limited capacity). These disjunct links can be encoded in the constituency-matrix C which limits the control decisions through $Cv_t \leq c$ (with $\mathbf{1}$ being the vector of ones, with appropriate dimension). This means, that the controller is usually left with the decision of which information to route first. While this system-model is mostly in alignment to the usual one [2], we now introduce some novel features, that aim to model especially wireless communication more accurately.

(F1) We allow our links (columns of R) to have discrete values (in contrast to the usual binary ones). This allows information of different type to have different size or different importance since high entries in a communication link will mean faster transportation.

(F2) We allow each column of R to have multiple positive and negative entries and define that we can *not* activate a link, when (due to the system evolution) at least one entry of q_t would become negative. The second part is equal to the control restriction

$$v_t \in \mathcal{V}_t := \big\{ v_t \in \{0,1\}^m \ : \ Cv_t \le c \ , \ q_t + R^{-} v_t \ge 0 \big\} \tag{1.6}$$

where both inequalities are meant element wise and R^{-} equals R without its positive entries. This makes it possible to couple the processing of information, or model a demand [5, p.273] by forcing certain buffers to only be depletable together. Especially the last inequality in (1.6) deviates from the standard models used in [2] and [5], and is responsible for many results that are presented here.

(F3) As already done in [2], we capture the short term wireless characteristics (e.g. channel fading), by introducing a diagonal weight matrix $M \in [0,1]^{n_v \times n_v}$ which encodes the success probability of an activated communication link. Hence a controller might activate some link in time-step t, which in the end will not influence the system due to the communication being unsuccessful. More specifically, we define that the controller has knowledge of all communication links (now encoded as columns of the matrix R) and their success probabilities (encoded in M). For the real system evolution however, we evaluate M by performing Bernoulli trials (coin tosses) on its individual entries. Specifically,

$$\text{Bern}[\xi] = \begin{cases} 1 & \text{w.p. } \xi \\ 0 & \text{otherwise} \end{cases} \tag{1.7}$$

for any $\xi \in [0,1]$. Using this operator on matrices means using it separately on its individual entries. We then obtain M_t from (1.5) through $M_t = \text{Bern}[M]$. As a result, some columns of RM_t are set to 0, whereas others are as in R, independent of the decisions made by the controller.

(F4) We capture long term wireless characteristics (e.g. entities leaving the entire system), by having a whole set of probability matrices such that $M \in \{W^i\}, i = 1 \dots n_s$ and choosing one of these through an underlying discrete-time Markov chain (DTMC)

$$s_t = \text{DTMC}\left(\{1 \dots n_s\}, P, s_0\right) \tag{1.8}$$

on its index set $\{1 \dots n_s\}$ with left side transition matrix P and initial state s_0. In each time-step this DTMC will dictate, which W^i to use for the description in (F3). We can therefore write $M_t = \text{Bern}[W^{s_t}]$. This setup is tightly related to *discrete-time Markov jump linear systems* [10].

1.2.3 Predictive Network Control (PNC)

We now introduce our new control policy, called Predictive Network Control (PNC). Like for any policy, the purpose of PNC is to let the system process as much information per time as possible, i.e. to escort it out of the system.

As mentioned, PNC is inspired by the paradigms of Model-Predictive Control (MPC) [9]. Therefore we first define a prediction-model, where quantities are labeled with a dot on top $(\dot{q}_t, \dot{v}_t)$ so as to distinguish them from the actual system-model (there is no connection to the time derivative). A cost function $J(\cdot)$ then assigns a cost to a whole trajectory of control-vectors $\tilde{\dot{v}}_t = \begin{pmatrix}\dot{v}_t^\intercal & \dot{v}_{t+1}^\intercal & \dots & \dot{v}_{t+H-1}^\intercal\end{pmatrix}^\intercal$. We call the length of the trajectory the *prediction horizon* H. Going on, we calculate the minimizing trajectory of input-vectors $\tilde{\dot{v}}_t^*$ and intuitively apply its first vector $\dot{v}_t^*$ to our system to advance to the next time-step $t+1$. However once there, we do *not* apply the second vector of the optimal trajectory, which would be $\dot{v}_{t+1}^*$, since it is outdated. We instead repeat the whole process (minimizing, applying the first input-vector of the optimal trajectory) and thus obtain an implicit feedback control law.

In what follows, we will assume that the current time-step is $t = 0$. We choose the cost $J(\cdot)$ to be of quadratic form

$$J(\tilde{\dot{v}}_0, q_0, s_0) = \mathbb{E}\left[\sum_{t=1}^{H} \dot{q}_t^\intercal Q_q \dot{q}_t + \dot{v}_{t-1}^\intercal Q_v \dot{v}_{t-1} \,\middle|\, q_0, s_0\right] \tag{1.9}$$

with Q_q and Q_v being symmetric, positive definite matrices. Naturally, we do not have a choice but to work with the expectation $\mathbb{E}[\cdot]$ of future queue-vectors due to the stochastic system evolution. For this cost function, we can find the optimal, minimizing control $\tilde{\dot{v}}_0^*$ by transforming the problem into a standard quadratic program. The rest of this section shows how this can be accomplished (despite of the stochastics in the system evolution).

To handle the DTMC, it helps to define expanded versions of s_t, P, and R as

$$\tilde{e}_{s_t} = e_{s_t} \otimes I_n \qquad \tilde{P} = P \otimes I_n \qquad \tilde{R} = \begin{pmatrix} RW^1 \\ \vdots \\ RW^{n_s} \end{pmatrix} \tag{1.10}$$

where e_{s_t} is the s_t-th unit-vector (which is 1 at the (s_t)-th component and 0 everywhere else). With this we can express the expected future routing-matrix given an initial Markov-state as

$$\bar{R}_t(s_0) := \mathbb{E}[RM_t \,|\, s_0] = \left(\tilde{P}^\intercal \tilde{e}_{s_0}\right)^\intercal \tilde{R} \tag{1.11}$$

It follows that the expected queue-vector becomes

$$\mathbb{E}[q_t \,|\, q_0, s_0] = q_0 + \sum_{i=0}^{t-1} \bar{R}_i(s_0)\dot{v}_i + t\bar{a} \tag{1.12}$$

Substituting this into the cost function yields three cost terms, which are constant, linear and quadratic with respect to our control:

$$J(\tilde{\dot{v}}_0, q_0, s_0) = J_\mathrm{c}(q_0, s_0) + J_\mathrm{l}(q_0, s_0)\tilde{\dot{v}}_0 + \tilde{\dot{v}}_0^\intercal J_\mathrm{q}(q_0, s_0)\tilde{\dot{v}}_0 \tag{1.13}$$

The first term is independent of $\dot{v}_t$ and therefore without concern to us. The linear term can be expressed with (1.11) as

$$J_\mathrm{l}(q_0, s_0) = q_0^\intercal 2Q_q \begin{pmatrix} (H-0)\,\bar{R}_0^\intercal(s_0) \\ (H-1)\,\bar{R}_1^\intercal(s_0) \\ \vdots \\ (H-H)\,\bar{R}_{H-1}^\intercal(s_0) \end{pmatrix}^\intercal + \bar{a}^\intercal R \begin{pmatrix} (H+1)\,(H-0)\,\bar{R}_0^\intercal(s_0) \\ (H+2)\,(H-1)\,\bar{R}_1^\intercal(s_0) \\ \vdots \\ (H+H)\,(\,1\,)\,\bar{R}_{H-1}^\intercal(s_0) \end{pmatrix}^\intercal \tag{1.14}$$

For the quadratic cost term, we need an expression for the expected square of the system-matrices. It is easy to verify that for $k \geq l$

$$\bar{\bar{R}}_{k,l}(s_0) := \mathbb{E}\big[M^{k\intercal} R^\intercal Q R M^l \,\big|\, s_0\big] = \Big(\tilde{P}^l \tilde{e}_{s_0}\Big)^\intercal \begin{pmatrix} \tilde{R}^\intercal \tilde{P}^{k-l} \tilde{e}_1 Q_q R W^1 \\ \tilde{R}^\intercal \tilde{P}^{k-l} \tilde{e}_2 Q_q R W^2 \\ \vdots \\ \tilde{R}^\intercal \tilde{P}^{k-l} \tilde{e}_{n_s} Q_q R W^{n_s} \end{pmatrix} \tag{1.15}$$

It follows that

$$J_\mathrm{q}(q_0, s_0) = J_\mathrm{q}(s_0) = (I_H \otimes Q_v) + \begin{pmatrix} H \cdot \bar{\bar{R}}_{0,0}(s_0) & (H-1) \cdot \bar{\bar{R}}_{0,1}(s_0) & \dots \\ (H-1) \cdot \bar{\bar{R}}_{1,0}(s_0) & (H-1) \cdot \bar{\bar{R}}_{1,1}(s_0) & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{1.16}$$

Furthermore, we need to handle the following 3 constraints:

(i) The binarity of $\tilde{\tilde{v}}_0$

$$\tilde{\tilde{v}}_0 \in \{0,1\}^{m \cdot H} =: \mathcal{B} \tag{1.17}$$

(ii) The constituency of $\tilde{\tilde{v}}_0$

$$[I_H \otimes C]\, \tilde{\tilde{v}}_0 \leq \mathbf{1} \tag{1.18}$$

(iii) The constraint of $\dot{q}_t \in \mathbb{N}^{n_q}$. Here, the discreteness is provided by the system-model (1.5). However, there are several ways to translate the positiveness into the optimization. Simply forcing $\dot{q}_t \geq 0$ for $t = 1 \dots H$, resulting in H so-called *hard constraints*, is most conservative and neglects any information on the arrival rate (setting it to the worst case, which is 0). Indeed, it would suffice to only force positiveness for the first evolution, $\dot{q}_1 \geq 0$, since this alone would already guarantee overall positiveness due to the repeated application of the optimization. The rest of the constraints could then be reformulated as *soft constraints* $\mathbb{E}[\dot{q}_t] \geq 0$ for $t = 2 \dots H$. We suggest an adjustable approach, depending on the variance of the arrival. The more evenly the arrival, the more soft constraints should be used. This should improve performance due to a better prediction of the future states of the network. Using only one hard constraint for the first two steps yields the following constraint

$$\underbrace{\begin{pmatrix} R^- & 0 & 0 & \dots & \\ R & R^- & 0 & \dots & \\ \bar{R}_0(s_0) & \bar{R}_1(s_0) & \bar{R}_2^-(s_0) & & \\ \vdots & \vdots & & \ddots & \\ \bar{R}_0(s_0) & \bar{R}_1(s_0) & \bar{R}_2(s_0) & \dots & \bar{R}_{H-1}^-(s_0) \end{pmatrix}}_{D(s_0)} \tilde{\tilde{v}}_0 \leq \underbrace{\begin{pmatrix} q_0 \\ q_0 \\ q_0 + 3\bar{a} \\ \vdots \\ q_0 + H\bar{a} \end{pmatrix}}_{d(q_0, \bar{a})} \tag{1.19}$$

Multiple hard constraints up to time-step τ can be implemented by replacing any expected system-matrices matrices with R on the LHS and removing any $\bar{a}$ term on the RHS of the constraint up until row τ. Note that the last matrix in each row is denoted with $(\cdot)^-$. This operator transforms the matrix by setting every positive entry of it to 0. Let A be any real valued matrix and

a_{ij} its entries, then $a_{ij}^{-} := \min\{a_{ij}, 0\}$. This is necessary to forbid the system to route a single packet of information through multiple queues in a single time-step.

Finally, we can define the PNC as the policy that in each time-step solves the binary quadratic program

$$\begin{gathered}\phi^{\text{PNC}}(q_0, s_0) := \dot{v}_0^* = \operatorname{farg}\min_{\tilde{v}_0} \; J_{\text{l}}(q_0, s_0)\tilde{v}_0 + \tilde{v}_0^\intercal J_{\text{q}}(s_0)\tilde{v}_0 \\ \text{s.t.} \\ \tilde{v}_0 \in \mathcal{B}, \quad [I_H \otimes C]\,\tilde{v}_0 \leq \mathbf{1}, \quad D(s_0)\tilde{v}_0 \leq d(q_0, \bar{a})\end{gathered} \tag{1.20}$$

and initializes the first optimal control $\dot{v}_0^*$ from its solution (i.e. the first argument, farg). We implicitly assume, that the Markov-state s_0 is known together with all other used parameters.

1.2.4 Simulations

In what follows, we showcase the behavior of two exemplary networks, when controlled by the PNC policy. We compare it directly with the MW policy, which as mentioned is often used as a benchmark. Note however, that especially feature (F2) makes MW deviate from its usual throughput optimal behavior. Also, since the arrival has stochastic character, each simulation yields slightly different results. The here presented graphs are therefore only representatives which, to the best of our knowledge, do showcase the usual behavior of the quantity in question.

Note, that PNC, as defined in Section 1.2.3, uses a binary quadratic optimization over a control variable of potentially high dimension (depending on horizon H). Whereas MW uses binary linear optimization. Thus any gain in performance has to be set into relation to the additional computational cost. Having this in mind, we developed a modified version of PNC, called linear PNC (L-PNC), which does neglect the quadratic term in the optimization (setting $J_{\text{q}}(s0) = 0$ in (1.20)). To separate both versions, we will from now on call the standard PNC (as introduced in Section 1.2.3) quadratic PNC (Q-PNC). Indeed, all simulations show little to no change in performance when using L-PNC instead of Q-PNC, though this is still subject to research. Finally, since MW does not consider any direct cost contributed to the control-vector, we choose $Q_q = I_n$ and $Q_v = 0$ to be able to compare the policies.

Generic Example

We first consider an example, specifically constructed to showcase the mechanism through which PNC dominates MW. This example does *not* need any probability weights, as introduced in (F3) and (F4), but only makes use of (F1) and (F2). Specifically we look at two queues, $q^{[1]}$ and $q^{[2]}$, who are subject to arrival rates $\bar{a}^{[1]}$ and 0, respectively. As shown in Figure 1.2, the controller is in every time-step presented with the same three *mutually exclusive* options (links), decoded as columns in the constant routing-matrix

$$R = \begin{pmatrix} -2 & -1 & -5 \\ 0 & 1 & -1 \end{pmatrix} \tag{1.21}$$

The first column directly decreases $q^{[1]}$ while not changing $q^{[2]}$. This can be interpreted as the data of $q^{[1]}$ being processed and afterwards leaving the system. The second column also decreases $q^{[1]}$ for the price of increasing $q^{[2]}$ which models a transmission from $q^{[1]}$ to $q^{[2]}$. The third column allows the controller to heavily decrease $q^{[1]}$; however according to (F2) it can only take this action if $q^{[2]}$ is nonempty. An interpretation could be, that the parallel processing of the information is extremely beneficial (e.g. due to a lack of storage).

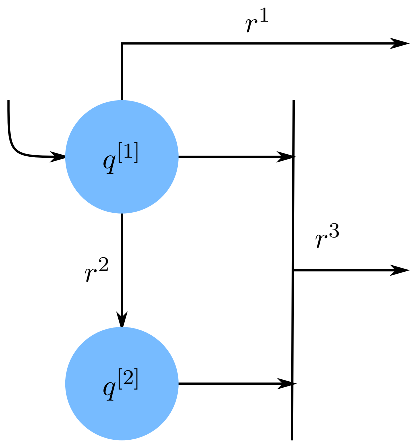


Fig. 1.2: Queueing model of the generic example. Packets only arrive at queue $q^{[1]}$ and can only leave the system via the links r^1 and r^3.

For this specific case, we designed link 1 in such a way that MW will almost always prefer it compared to link 2. We say that link 1 *dominates* link 2 under the MW policy. As a consequence, MW will never be able to use link 3 (since there will not be any information in $q^{[2]}$). Hence the maximal arrival rate that MW can handle is $\bar{a}^{[1]} = -R^{[1,1]} = 2$.

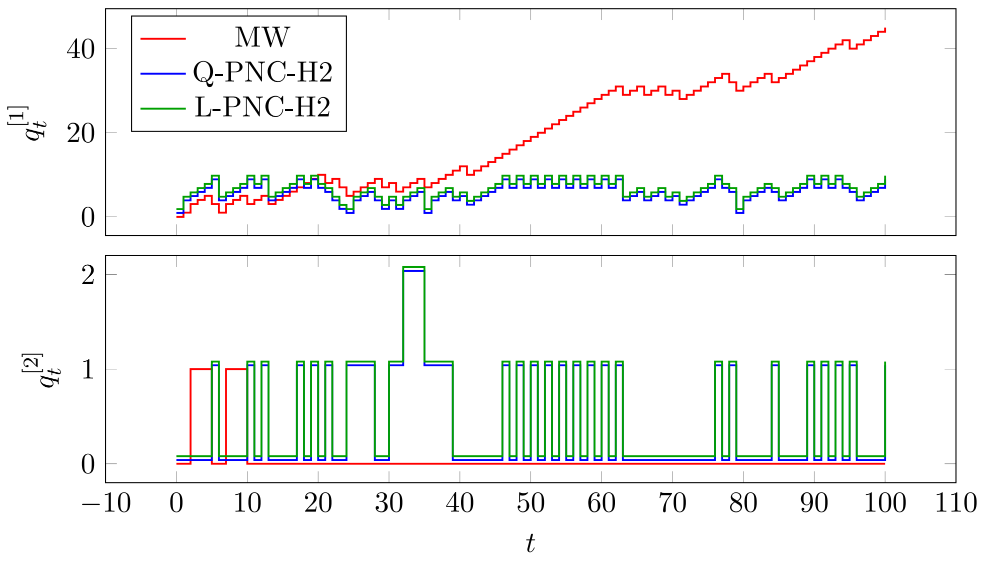


Fig. 1.3: Queue-states $q_t^{[1]}$ and $q_t^{[2]}$ over time for the generic example for 3 different policies. The MW policy does not stabilize the system under the specific arrival process.

A superior strategy would be to switch periodically between link 2 and 3, enabling the system to be stable under a maximal arrival rate of $\bar{a}^{[1]} = -\frac{1}{2}R^{[1,2]} - \frac{1}{2}R^{[1,3]} = 3$.

PNC, with a horizon of at least $H = 2$ indeed follows this strategy when possible (the stochastical character of a_t may prevent it from time to time). Through this behavior, PNC does stabilize the Network for a wider set of arrival rates. Figure 1.3 compares the queue-states over the first 100 time-steps for the mentioned policies. In this case, we chose $\bar{a}^{[1]} = 2.4$ where we simulated a_t as a Bernoulli trial (coin toss) with probability of 0.8 and a weight of 3. One can clearly see the growing queue length under the MW policy, whereas any of the PNC policies does result in a stable behavior. In other words: PNC can handle a larger network load than MW.

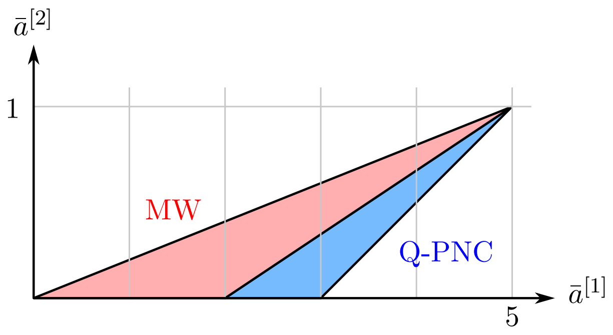


Fig. 1.4: Stability region for different policies for the generic example. While MW only stabilizes the red region, Q-PNC stabilizes both the red and the blue region which can be shown to be the largest possibly stabilizable region.

The whole stability region for MW is shown in Figure 1.4 as the red area. PNC, implemented with full hard constraints, does expand onto this with the blue area (but also still stabilizes the red one). For this specific example, we do not increase the stability region if we chose $H > 2$. However, there do exist such networks where the stability region increases with increasing horizon. E.g. changing the $\begin{pmatrix}-5 & -1\end{pmatrix}$ column in the R matrix to $\begin{pmatrix}-5 & -2\end{pmatrix}$ would be such a network.

Natural Example

The second example, that is showcased in the following, is obtained by modeling a rather real world scenario. We consider two *users*, interacting through a mobile game. Additionally a *game master* is also needed to provide neutral information to both parties as shown in Figure 1.5. The game consists of many turns, each one progressing according to the following scheme: The game master sends information to both users; the game waits until both users interact with this information; their inputs are communicated between each other and evaluated; the turn ends. This network can be modeled with the help of three queues: $q^{[1]}$ holds the information, the game master sends in the beginning of each turn; $q^{[2]}$ holds the information that was successfully send to both users but is not yet processed due to missing interactions by the users; $q^{[3]}$ symbolizes the inputs of the users and is increased, only when both users did interact with the information from $q^{[2]}$. Hence, information can only exit the system, when both $q^{[2]}$ and $q^{[3]}$ are non empty, i.e. only when the game master did successfully send information and *both* users interacted with it. We further define two communication channels: one between the users and one from game master to both users. We assume that both channels are mutually exclusive, so that in each time step, a policy has to decide for one of them to be inactive. Figure 1.6 shows the model of the network with arrivals (red) and communication links (blue).

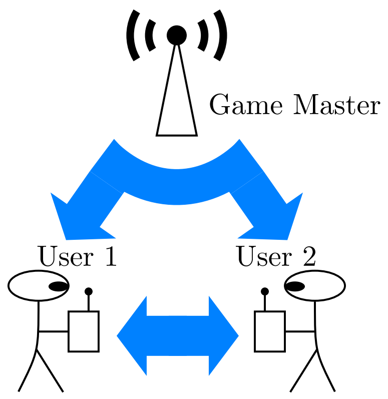


Fig. 1.5: Set-up for the natural example, depicting a game played by two users.

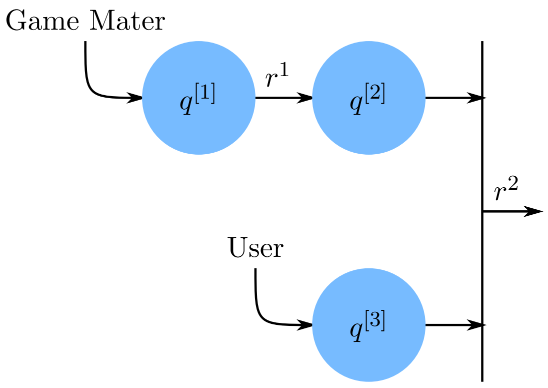


Fig. 1.6: Queueing model of the natural example. Packets can only leave the system if the users ($q^{[2]}$ and $q^{[3]}$) "cooperate".

Note that $q^{[2]}$ acts as a buffer, which, when filled, allows for $q^{[3]}$ to be decreased. The superiority of PNC over MW is based on the utilization of that buffer. Through prediction of the Markov chain, PNC usually keeps this buffer at a higher level. MW on the other hand tends to use this buffer to decrease $q^{[3]}$ every time it gets the chance to do so, which is not the optimal strategy.

In this simulation example, we model the wireless characteristics according to (F3) and (F4) by introducing a *good* network state, in which both communication channels are guaranteed to work, and a *bad* network state, in which only the communication between the users is possible. Both are represented by the weight matrix W^{good} and W^{bad} respectively. We use the following set of parameters where we again model the stochastics of a_t as Bernoulli trials:

$$\begin{gathered}\bar{a} = \begin{pmatrix}0.5 & 0 & 0.9\end{pmatrix}^\intercal, \quad C = \begin{pmatrix}1 & 1\end{pmatrix}, \\ R = \begin{pmatrix}-3 & 0 \\ 3 & -1 \\ 0 & -1\end{pmatrix}, \quad P = \begin{pmatrix}0.1 & 0.2 \\ 0.9 & 0.8\end{pmatrix}, \\ W^1 = W^{\text{good}} = \begin{pmatrix}1 & 0 \\ 0 & 1\end{pmatrix}, \quad W^2 = W^{\text{bad}} = \begin{pmatrix}0 & 0 \\ 0 & 1\end{pmatrix}\end{gathered} \tag{1.22}$$

Figure 1.7 shows $q^{[1]}$ over time, which is an indicator for the speed of the game. It therefore can be interpreted as a performance measure. We can see, that Q-PNC outperforms MW already for $H = 2$. For $H = 1$, both policies exhibit the same behavior (which as mentioned is an indication that the quadratic part of the optimization does not influence the result significantly). It should be noted, that both policies cannot stabilize the system for the specified arrival rate and that we used soft constraint for all but the first evolution of the system in the PNC algorithm.

For a stabilizable arrival rate Figure 1.8 shows the performance gains that Q-PNC with $H = 3$ yields compared to MW. The average queue-states are roughly about 10% smaller for Q-PNC. We omit $q^{[2]}$ which is usually higher for PNC, since it is meant to act like a storage and thus does not give any additional indication on the performance. For these graphs, we switch between periods of high and low

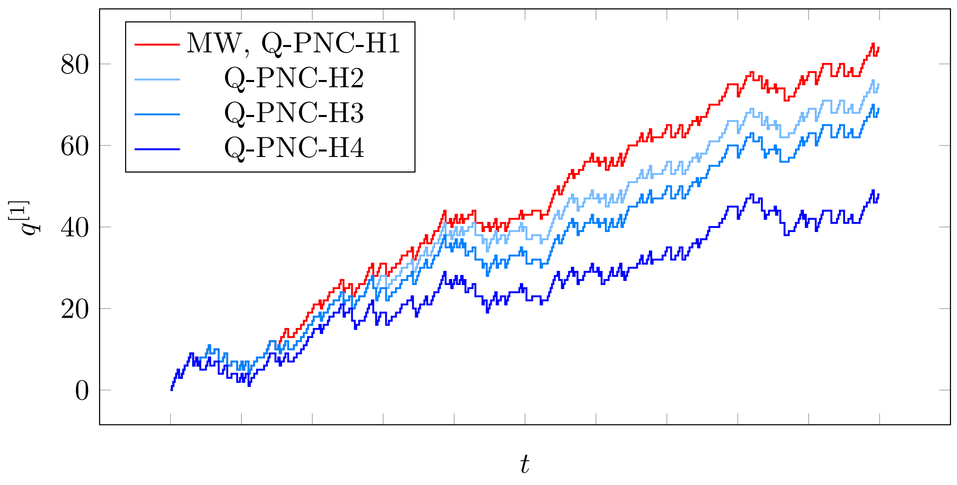


Fig. 1.7: The effect of larger prediction horizons H on the throughput performance. Though no policy can stabilize the specific arrival, predictive control policies facilitate a larger throughput.

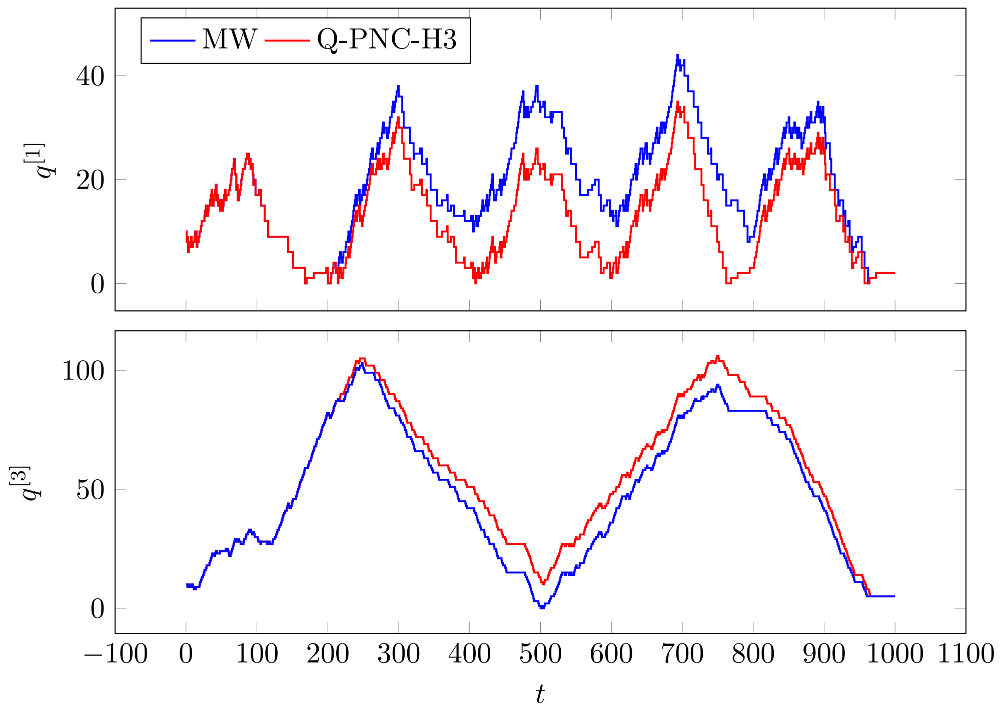


Fig. 1.8: Queue-states over time for a fluctuating arrival. Since the fluctuation prevents MW to reach its steady state, Q-PNC-H3 manages to maintain lower queue-backlogs.

arrivals. Specifically we used $\bar{a}^{[1]} = 0.375 \pm 0.3$ and $\bar{a}^{[3]} = 0.38 \pm 0.38$ for intervals $\Delta t^{[1]} = 100$ and $\Delta t^{[3]} = 250$ respectively; the arrivals were initialized as Bernoulli trials.

1.2.5 Conclusion

Our new control policy seems to outperform the standard MW policy when the network is based on our more general system-model. Especially the gain in stability region is a huge and surprising advantage. Another conclusion, that might be overlooked when reading this paper, is the following negative indication: for rather simple networks (e.g. broadcasting scenarios), our policy does *not* lead to any significant gains. After the initial results presented here, our immediate research will evolve around the points of network classification, performance to cost trade-off and prove of stability for our new control policy.

1.3 Concluding remarks

An important issue that the paper misses to address (but which is rigorously discussed in Paper 2) is the implicit dependence of the control variable $\dot{v}_t$ on future queue-states and future Markov-states. Specifically, in (1.9), $\dot{v}_t$ could depend on $\dot{q}_t$ and also s_t. As this is the prediction-model and not the actual system-model, we are free to choose the complexity of the dependencies of $\dot{v}_t$. The derivations in the paper implicitly assume that $\dot{v}_t$ is insensitive to both the current DTMC state s_t and the corresponding queue-state $\dot{q}_t$. If this was not the case and $\dot{v}_t$ would be treated as a function of s_t and $\dot{q}_t$, the complexity of the optimization would grow significantly. To elaborate we refer to Figure 1.9: on the left side, at some initial time-step $t = 0$, a single control variable is employed for the optimization. Due to the stochastics in the system, this could lead to various possible system-states in time-step $t = 1$. For each of these possible system-states, we have to consider a separate control-vector which, if applied, would lead again to various resulting system-states. Staying true to the figure, we would have to employ one control-vector for time-step $t = 0$, two control-vectors for $t = 1$, four control-vectors for $t = 2$ and so on, resulting in $2^H - 1$ variable control-vectors (H being the prediction horizon) over which the optimization has to find the optimal trajectory. In contrast, on the right side of Figure 1.9, the various states that could possibly be reached from $t = 0$ with the employed control-vector are immediately condensed by considering their probability of occurrence and taking the expectation (blue state). Using this expectation as a new starting point, only one additional control-vector has to be employed for the prediction in time-step $t = 1$. This way we end up with H variable control-vectors over which the optimization is performed. Obviously, we have a much reduced complexity for the price of possibly sub-optimal solutions to the actual optimization problem.

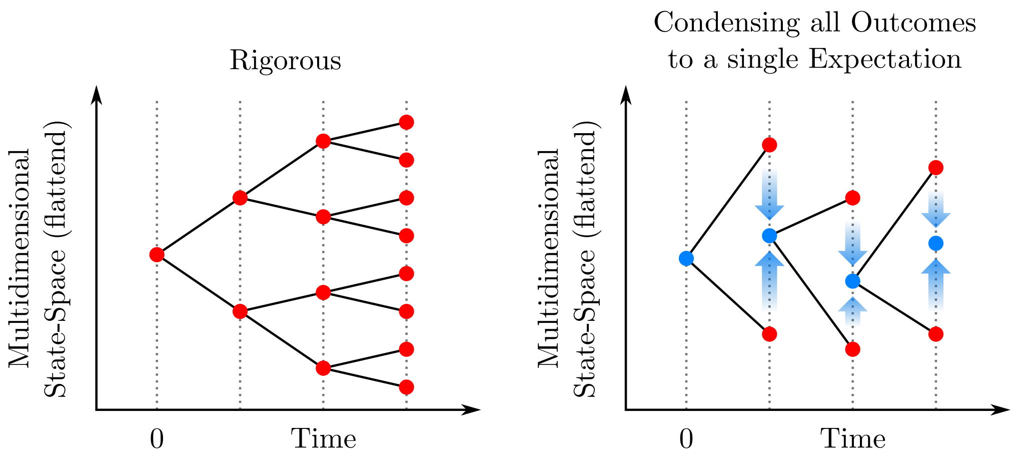

Rigorous
Multidimensional State-Space (flattend)
0
Time
Condensing all Outcomes to a single Expectation
Multidimensional State-Space (flattend)
0
Time

Fig. 1.9: Complexity in the prediction-model. Conditioning future control-vectors on all possible future realizations (in a causal way) causes the complexity to grow exponentially.

Compared to the original version of the paper, Figure 1.3 has been updated as there was a minor mistake in the code of the simulation in the published version. Crucially, the main statement stays the same, namely that MW does not and PNC does stabilize the system. However, contrary to what is stated in Section 1.2.4, MW *can* activate the second link (which shifts a packet from $q^{[1]}$ to $q^{[2]}$, see Figure 1.2), but will lose this ability as time progresses almost surely. This is visualized in Figure 1.10 which plots the negative control-options into the state-space as introduced in Figure 1.1. The colored regions of the state-space map a state vector to the similarly colored control-option (under the MW policy). If the state vector lies in the non-colored area of the state-space, the chosen control-option is the 0 vector (idling). The orange arrows indicate a possible evolution of the state vector. Starting from 0 the arrival could increase the x-component by 3 repeatedly, while the chosen control-option (u^1) decreases the same component by 2. Given that the average arrival is $0.8 \cdot 3 = 2.4$, this yields an evolution that lets the state vector remain on the x-axis and grow to infinity almost surely since the network controller will stick to control-option u^1. For small queue-states, there is the possibility that control-options u^2 and u^3 are used instead, which could compensate the influx completely since an alternating activation of u^2 and u^3 yields an efflux in the x-component by $-\frac{+1-5}{2} = 2.5$ (which is larger than the average arrival with of 2.4). However, even if this alternating activation would be invoked by the MW policy, the state vector would return completely to the origin with finite probability. And starting from the origin, there is always a probability of the aforementioned violet-colored evolution to occur which, all things considered, almost surely leads to an unstable evolution.

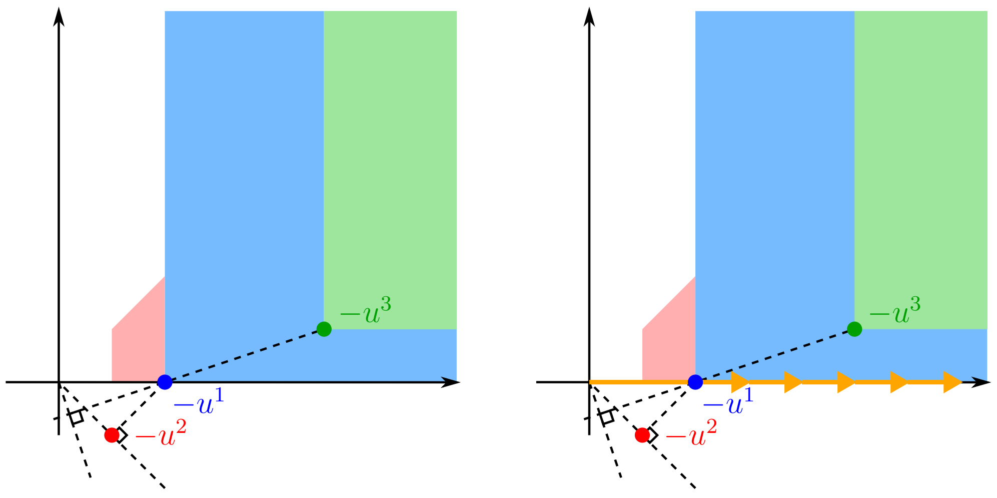


Fig. 1.10: Visualization of the "Natural Example" from Section 1.2.4. While there exists the possibility that MW uses control-options u^2 and u^3 (if the queue-state lies in the red or green area), almost surely the queue-state will be located on the x-axis resulting in an unstable behavior since there, MW can only activate u^1.

Paper 2

Throughput Optimality of PNC

Full Paper Title Model-Predictive Control for Discrete-Time Queueing Networks with Varying Topology

Publisher IEEE Transactions on Control of Network Systems (TCNS), 2021

Authors Richard Schöffauer, Gerhard Wunder

Acknowledgment This work is funded by the DFG Priority Program 1914.

Abstract In this paper, we equip the conventional discrete-time queueing network with a Markovian input process which, in addition to the usual short-term stochastics, governs the mid- to long-term behavior of the links between the network nodes. This is reminiscent of so-called *Jump-Markov Systems* in control theory, allows the network topology to change over time, and thus facilitates to model a plethora of useful applications in wireless communication, traffic, or logistics. We argue that the common back-pressure control policy is inadequate to control such network dynamics and propose a novel control policy inspired by the paradigms of *Model-Predictive Control.* Specifically, by defining a suitable but arbitrary prediction horizon, our policy takes into account future network states and possible control actions. This stands in clear contrast to most other policies which are myopic, i.e. only consider the immediate next state. We show numerically that such an approach can significantly improve the control performance and introduce several variants of the policy, thereby trading off performance versus computational complexity. In addition, we prove so-called *throughput optimality* of our policy which guarantees stability for all network flows that can be maintained by the network topology. Interestingly, in contrast to general stability proofs in Model-Predictive Control, our proof does not require the assumption of a terminal set, i.e. the prediction horizon is *not* required to be large enough as to reach a predetermined set of states with special properties. Finally, we provide several illustrating examples, one of which being a network of assembly-queues. This network in particular constitutes an interesting system class for which our policy exerts superiority over general back-pressure policies, with the latter ones even losing their throughput optimality.

2.1 Preliminary Remarks

This paper revisits the PNC policy from Paper 1, this time however in a much more rigorous fashion. The main contribution is the proof of the throughput optimality of PNC which is made very intricate by the repeated application of the policy's internal optimization in each time-step, thereby discarding the entire tale of the previously calculated optimal trajectory.

From the side of queueing networks, throughput optimality is usually proven by employing a Lyapunov function over the state-space and showing that the evolution of the network (governed by the policy in question) does, in expectation, drift downwards this Lyapunov function, with the exception of a finite set of states for which this must not be the case. In PNC, the first control-vector of the optimal trajectory (i.e. the control-vector that is actually applied to the system) might not be optimal by itself for the first state transition. On the other hand, the rest of the optimal trajectory is discarded when the optimization begins again in the next time-step. Hence, it is not immediately clear how to show that the PNC policy guides the system-state downward some suitable Lyapunov function.

On the other hand, techniques from the realm of Model-Predictive Control generally require existence of a terminal set of system-states which then must be reached by at least the last predicted state (in order to guarantee stability). In this terminal set, the system dynamics are assumed to be manageable (e.g. by linearization around an operating point) so that the system would never leave the terminal set, once it is reached. Obviously, such a terminal set does not exist in our queueing networks, because the system dynamics cannot be simplified depending on the system-state. In this paper, however, we manage to find a way to prove throughput optimality for PNC by applying a specially adopted Lyapunov function.

2.2 Paper Body

2.2.1 Introduction and Related Research

Discrete-time queueing networks are used to model a variety of scenarios, ranging from traffic control over parallel computing to wireless communication. They are closely related to the canonical discrete-time controlled system

$$x_{t+1} = Ax_t + B_t v_t + D_t w_t \tag{2.1}$$

with x_t, v_t, w_t being state-, control-, and disturbance-vector, and t designating the time-step. However, they feature some significant differences: (i) The controls v_t are binary in nature and linearly constrained, e.g. due to the interference properties of wireless channels. (ii) The state resides on the discrete set $x_t \in \mathbb{N}^n$ where it exhibits no internal dynamics (A is an identity matrix). (iii) The matrices B_t (and D_t) behave stochastically, implying that the effect of a control decision is not certain. Together with the class of back-pressure control policies, those systems form a well investigated subclass of control problems.

The prototype back-pressure policy, that we will call the MaxWeight policy (MW), was first introduced in [2] where the authors also proved its much celebrated property of being *throughput optimal.* This means that MW can stabilize the queue-backlog for any load of arriving packets/customers that is sustainable under the network topology. Over time, many variations of MW where developed, e.g. to allow for a generalized control objective [3, 4], or to increase its performance in special cases like networks with input-queued switches or time-varying channels [11, 12]. Specific shortcomings of MW, like high end-to-end delay were investigated in [13, 6, 14] and later partially remedied by [12, 14, 15, 16], using e.g. shortest path algorithms to reduce delay especially in low traffic scenarios.

In this paper, we propose a novel control policy that is predictive in nature and therefore named Predictive Network Control (PNC). It can be regarded as a generalization of MW since it contains MW as a special case. However, while MW and all its derivations are *myopic*, i.e. only aim to improve the system-state for the *immediate* next time-step, PNC aims to improve the system-state for *multiple* time-steps up until a prediction horizon. This leads to the calculation of an entire optimal *trajectory* of control-vectors. Then, instead of applying the entire trajectory for the next few time-steps, only the first control-vector is applied to the system and the process repeats in the consecutive time-step. This allows the controller to react to any unforeseen changes in the control system [9].

Our PNC policy can be categorized as Model-Predictive Control (MPC) that is tailored specifically towards queueing networks. MPC is a well-established branch of control theory and can cope very easily with hard constraints and non-linearities, making it particularly suited for our control problem. However, its advantages come at the cost of increased requirements on computational resources. So far, there has only been one attempt to bring MPC to queueing networks: In [17] the authors focus on a special case of the standard model in which only the arrivals to the system are of stochastic nature. Their investigations are limited to numerical simulations which show better system performance (less outliers over time in the queue-backlog) for a designed MPC controller compared to simple feedback control laws. Since our queueing network will include a much higher degree of stochastics, we will not follow up on their work.

Concerning the stochastics, we let B_t (the matrix responsible for the topology and the quality of the links between the nodes of the network) be governed by a discrete-time Markov chain (DTMC) on top of a Bernoulli trial. This allows to model long-term and short-term effects, respectively. Take e.g. wireless relay networks with user mobility: Here, short-term interference leading to packet loss can be modeled by the Bernoulli trial, while long-term change in channel quality due to the mobility can be expressed by the DTMC [18]. The proposed stochastics together with the employed system-model also fit very well to logistic and traffic scenarios. For the former, take parcel shipping companies: the Bernoulli trial models the probability of costumers not being home to receive the parcels and the DTMC governs those probabilities according to time of day or day of the week. For the latter one, take public transportation services: the Bernoulli trial models passengers arriving on time to catch a connecting service and the DTMC governs this probability based on general volume of traffic or weather conditions over the time of day.

Note that it is possible to merge DTMC and Bernoulli trial into an aggregate DTMC. However, doing so greatly diminishes the convenient separation between short- and long-term effects without significantly simplifying any mathematical derivations.

Control systems in which the model parameters change according to a DTMC are called Jump-Markov Systems (JMS). Since simple feedback controllers cannot detect this change, JMS are usually controlled with MPCs. There exist several control approaches regarding JMS, covering cases with linear [19, 20, 21] and even nonlinear system dynamics [22, 23] (the referenced works mainly differ in the choice of considered constraints). However, all these works deal with conventional control systems, where the controller usually tries to follow a reference trajectory and the first moment of the disturbance is zero. In contrast, from the perspective of queueing networks, the disturbance represents the arrival of packets whose first moment is strictly positive, and the controller tries to maintain finite queues for any given arrival (hence, there is no need for a reference trajectory). For that reason, prior work on JMS is only partially applicable to our systems. To the best of our knowledge, we are the first to consider both JMS and MPC in the context of discrete-time queueing networks.

Specifically, our **contributions** are as follows: (i) We develop a JMS-adapted discrete-time queueing network and introduce a family of predictive control policies, based on the paradigms of MPC. (ii) We prove throughput optimality (the equivalent of stability) for these policies. (iii) We show the benefit of these policies over the conventional back-pressure control (MW) using numerical simulations. In particular, our policies seem to maintain their throughput optimality in networks with assembly-queues.

Eventually, we emphasize that our proof technique itself contains novel concepts and merges elements from the theory of queueing networks and MPC. On the one hand, conventional proofs *for MPCs* always assume a sufficiently long prediction horizon H so that a terminal set of system-states is reached. This terminal set has to be invariant under the MPC-induced control law and equipped with a separate cost function that has Lyapunov properties [9]. Opposed to that, our proof does *not* require a terminal set and therefore guarantees stability independent of the prediction horizon. On the other hand, conventional proofs *for queueing networks* typically try to find an upper bound for the *one-step* drift which readily works for *one-step* (myopic) policies [2, 3]. Opposed to that, we merge an H*-step* policy with a *one-step* drift, which is considerably more difficult and requires a completely new approach.

Notation: We will usually use the lower index to denote time-steps (x_t) and the upper index to count ($\mathcal{X} = \{x_t^1, \dots x_t^n\}$). If additionally the upper index is set in square brackets ($x_t^{[k]}$), then we count over elements of a vector or diagonal matrix. A stochastic process will be denoted with curly brackets ($\{x_t\}$).

2.2.2 System-Model & Prerequisites

System-Model

Our network consists of n_q nodes and n_v links. Node i ($i = 1, \dots n_q$) corresponds to a queue which holds $q_t^{[i]} \in \mathbb{N}$ packets in time-step t. The collection of these quantities is the *queue-vector* $q_t = (q_t^{[1]} \dots q_t^{[n_q]})^\intercal \in \mathcal{Q} = \mathbb{N}^{n_q}$.

Packets can be transmitted from one queue to another if there exists a directed link between the two and the link is activated. Link j ($j = 1, \dots n_v$) is represented by vector $r^j \in \{-1, 0, +1\}^{n_q}$ where negative and positive entries denote origin and destination queue, respectively. All links are collected as columns in the *routing-matrix* $R \in \{-1, 0, +1\}^{n_q \times n_v}$ which therefore holds the topology. Transmission of packets is expressed by summation: $q_t + r^j = q_t + Re^j$ (e^j being the j-th canonical unit vector).

The one-step evolution of our discrete-time queueing network can be expressed by

$$q_{t+1} = q_t + RM_t v_t + a_t \tag{2.2}$$

$$\text{subject to}$$

$$\begin{pmatrix} Cv_t \leq c \\ -R^- v_t \leq q_t \\ v_t \in \{0,1\}^{n_v} \end{pmatrix} \text{ and } \begin{pmatrix} M_t \sim \text{Bernoulli}(W^{s_t}) \\ W^{s_t} \in \{W^1, \dots W^{n_s}\} \\ \{s_t\} \sim \text{DTMC}(\{1, \dots n_s\}, P, s_0) \end{pmatrix}$$

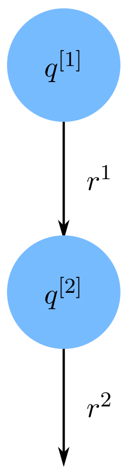
where the actual *system-state* is the pair (q_t, s_t) which is composed of the *queue-state* q_t and the *topology-state* s_t. The latter governs the stochastics of the links. Both $\{q_t\}$ and $\{s_t\}$ are observable.

Two antagonistic processes influence $\{q_t\}$: the efflux and the influx. The influx describes the arrival of new packets to the end of the queues and is denoted by $\{a_t\}$. We assume $\{a_t\}$ to be a sequence of i.i.d. upper bounded multivariate random variables $a_t \in \mathbb{N}^{n_q}$, $a_t \leq \hat{a}\mathbf{1}_{n_q}$, where $\mathbf{1}_{n_q}$ is the vector of ones with dimension n_q. We call $\bar{a} = \mathbb{E}[a_t]$ the *arrival rate*.

The efflux describes the departure of packets from the network (facilitated by links (r^j) that do not have a destination, i.e. a positive entry in their vectors). It is the purpose of the controller to manage this departure by routing and scheduling the packets through the network. For that purpose, in each time-step he may activate certain links via the binary *control-vector* $v_t \in \{0,1\}^{n_v}$ which is constrained by two effects: (i) The *constituency-constraints* $Cv_t \leq c$ prohibit simultaneous activation of certain links, e.g. due to interference. The dimensions of matrix C and vector c are case dependent, their entries are from the set $\mathbb{N}$. (ii) The *routing-constraints* $q_t + R^- v_t \geq 0$ ensure that a packet can only be scheduled for transmission if it is present at the corresponding queue, i.e. a packet may only traverse a single link per time-step instead of being routed through multiple links at once. Here, R^- is a copy of R without any positive entries such that transmitted packets do not arrive at their destination and simply disappear from the system. Hence, the controller can only schedule as many packets from a queue as are present at the start of the time-step t.

For illustration, we state topology and constituency matrices of a toy network in Figure 2.1 and derive the corresponding constraints: Considering only C and c, both components of v_t could be active simultaneously if $q_t = \begin{pmatrix} 1 & 0 \end{pmatrix}^\mathsf{T}$. However, if $q_t^{[2]} = 0$ it should not be possible to activate r^2 in time-step t. Thus, the routing-constraints are required.

Notably, there exists a network-inherent uncertainty: Scheduled transmissions may only succeed with a certain probability, leaving the network unchanged if they fail. We model this with two overlapping stochastic processes, a Bernoulli process and a DTMC. First, in time-step t, each vector r^j (link j) is multiplied by a random variable $m_t^{[j]} \in \{0,1\}$, belonging to the Bernoulli process $\{m_t^{[j]}\}$ with success-parameter $\bar{m}_t^{[j]} \in [0,1] \subset \mathbb{R}$. Collecting these values in the diagonal matrices $M_t = \text{diag}_{j=1,\dots n_v} \left\{ m_t^{[j]} \right\}$ and $\bar{M}_t = \text{diag}_{j=1,\dots n_v} \left\{ \bar{m}_t^{[j]} \right\}$ allows us to write the multiplication of links and random variables as RM_t. When we say that M_t is Bernoulli



$$\underbrace{\underbrace{R=\begin{pmatrix}-1&0\\+1&-1\end{pmatrix}}_{\begin{pmatrix}1&0\\0&1\end{pmatrix}v_t\le q_t}}_{\text{Routing-Constraints}} \qquad \underbrace{\underbrace{C=\begin{pmatrix}0&0\end{pmatrix}\qquad c=1}_{\begin{pmatrix}0&0\end{pmatrix}v_t\le 1}}_{\text{Constituency-Constraints}}$$

Fig. 2.1: Minimal example of a queueing network. The $+1$ entry from R is dropped in the routing constraints ($-R^{-}v_t \le q_t$) in order to prevent packets from traversing more than one queue per time-step.

distributed with success-parameter $\bar{M}_t$, $M_t \sim \text{Bernoulli}(\bar{M}_t)$, we mean that this is true for every element of the matrices separately. Note that the realization of M_t is not known to the controller in time-step t.

On top of that, we let the parameter matrix $\bar{M}_t$ change over time in order to model long-term effects in the connectivity of the network. To that end, we define a set of matrices $\mathcal{W} = \{W^1, \dots W^{n_s}\}$, with index set $\mathcal{S} = \{1, \dots n_s\}$. Each W^i is a diagonal matrix with dimension $n_v \times n_v$ and entries from $[0,1] \subset \mathbb{R}$. In each time-step, $\bar{M}_t$ is chosen to be from $\mathcal{W}$. This selection process is governed by the DTMC $\{s_t\} \sim \text{DTMC}(\mathcal{S}, P, s_0)$ where P and s_0 are transition matrix and initial state, respectively. Hence, $\bar{M}_t = W^{s_t}$. We assume that $\mathcal{W}$ and P are known and that $\{s_t\}$ is observable, finite and irreducible, and has stationary distribution $\pi = \pi P$.

Remark: In *conventional* networks, a link has exactly *one* origin. This implicit constraint is a prerequisite for all back-pressure policies to develop their throughput optimality. Though we will also use this constraint throughout the paper, our novel control policies seem to maintain their throughput optimality even when it is violated (see Section 2.2.5), allowing us to control networks with assembly-queues. Furthermore, our system-model also allows for multiple *distinguishable* types of packet *flows* by simply employing a block-diagonal structure in the system matrices and rearranging the constraints.

Control Policies and Throughput Optimality

In this subsection, we summarize the relevant control and stability concepts. Clearly, a control-vector v_t must be from the set

$$\mathcal{V} = \left\{ v \in \{0,1\}^{n_v} \;\middle|\; Cv \le c \right\} \tag{2.3}$$

Let $q_t \in \mathcal{Q}, s_t \in \mathcal{S}, \omega_t \in \Omega$ be the most recently observed realization from $\{q_t\}, \{s_t\}, \{\omega_t\}$, respectively, where $\{\omega_t\}$ is an *ergodic* stochastic process. Then we define a **control policy** as a mapping

$$\begin{aligned} \phi : \mathcal{Q} \times \mathcal{S} \times \Omega \to \mathcal{V}, \quad & (q_t, s_t, \omega_t) \mapsto \phi(q_t, s_t, \omega_t) = v_t \\ \text{s.t.} \qquad & -R^{-}\phi(q_t, s_t, \omega_t) \le q_t \end{aligned} \tag{2.4}$$

Hence, generally, we allow for randomized policies. It is readily verified that under such policies the system-process $\{q_t, s_t\}$ becomes a DTMC.

We say that a policy **stabilizes** a system for an arrival rate $\bar{a}$ if $\{q_t, s_t\}$ is positive-recurrent (i.e. has finite expected return time). Note that this implies finite expected system-states because in a single transition our system only allows for a finite set of follow-up states (and hence the queue-state can only change so much in a finite time).

Finally, we call a policy **throughput optimal** if it stabilizes a system for all arrival rates $\bar{a}$ for which there exists at least one (possibly unknown) stabilizing policy. This maximum set of arrival rates for which the system is somehow stabilizable is called the *stability region* $\mathcal{A}$. We can construct $\mathcal{A}$ as follows:

First, imagine all possible control policies that stabilize a given system. Under any such policy, $\{q_t, s_t\}$ is positive-recurrent. Also, $\{a_t\}$ is ergodic per definition. Therefore, the expectation of the efflux $RM_tv_t = q_{t+1} - q_t - a_t$ must, over time, converge to $-\bar{a}$. However, over time, the expectation of RM_tv_t must also be of the form $\sum_{\mathfrak{s}\in\mathcal{S}} \pi^{[\mathfrak{s}]} RW^{\mathfrak{s}}\bar{v}^{\mathfrak{s}}$. Here, $\pi^{[\mathfrak{s}]} = \mathbb{P}[s_t = \mathfrak{s}]$ is the probability of occurrence of topology-state $\mathfrak{s}$, $\bar{v}^{\mathfrak{s}} \in \mathrm{conv}(\mathcal{V})$ is the time-averaged control-vector generated by the policy in $\mathfrak{s}$, and $\mathrm{conv}(\cdot)$ is the convex hull. Together, we obtain the implication $\bar{a} \in \mathcal{A} \Rightarrow \exists \bar{v}^{\mathfrak{s}} \in \mathrm{conv}(\mathcal{V}) : \bar{a} + \sum_{\mathfrak{s}\in\mathcal{S}} \pi^{[\mathfrak{s}]} RW^{\mathfrak{s}}\bar{v}^{\mathfrak{s}} = 0$.

Second, imagine a policy that assigns to all states $(\mathfrak{q}, \mathfrak{s}) \in \mathcal{Q} \times \mathcal{S}$, that do not violate the routing-constraints (2.4), a random control-vector $\phi(\mathfrak{q}, \mathfrak{s}, \omega_t)$ with mean $\bar{v}^{\mathfrak{s}} \in \mathrm{conv}(\mathcal{V})$. For all other states, let the policy assign the same random control-vector but with some of its entries set to 0 so that the routing-constraints are met. Under this policy, we can prove positive-recurrence of $\{q_t, s_t\}$ for any arrival rate with $\bar{a} + \sum_{\mathfrak{s}\in\mathcal{S}} \pi^{[\mathfrak{s}]} RW^{\mathfrak{s}}\bar{v}^{\mathfrak{s}} < 0$. The proof is based on Foster's theorem [24] using the test-function $f(\mathfrak{q}, \mathfrak{s}) = \mathfrak{q}^\intercal\mathfrak{q}$. It can be performed analogously to the one in [2] but is much simpler since the just described policy is static. We obtain the implication $\exists \bar{v}^{\mathfrak{s}} \in \mathrm{conv}(\mathcal{V}) : \bar{a} + \sum_{\mathfrak{s}\in\mathcal{S}} \pi^{[\mathfrak{s}]} RW^{\mathfrak{s}}\bar{v}^{\mathfrak{s}} < 0 \Rightarrow \bar{a} \in \mathcal{A}$. Hence, concluding, a policy is **throughput optimal** if it stabilizes the system for every

$$\bar{a} \in \mathcal{A} = \left\{ \bar{a} \;\middle|\; \exists_{\mathfrak{s}\in\mathcal{S}} \bar{v}^{\mathfrak{s}} \in \mathrm{conv}(\mathcal{V}) : \bar{a} + \sum_{\mathfrak{s}\in\mathcal{S}} \pi^{[\mathfrak{s}]} RW^{\mathfrak{s}}\bar{v}^{\mathfrak{s}} < 0 \right\}. \tag{2.5}$$

The boundary of $\mathcal{A}$ typically is not part of $\mathcal{A}$ as can easily be verified by examining a network consisting of a single queue and a single link that drains this queue: An *optimal* policy would simply activate the link whenever possible, i.e. whenever the routing-constraints are not violated, no matter the arrival rate. Let the link have a success probability of 1 so that the closure of $\mathcal{A}$ is the interval $[0, 1]$. Let $\bar{a} = 1 \in [0, 1]$ with $\mathbb{P}[a_t = 0] = \mathbb{P}[a_t = 2] = \frac{1}{2}$. With this the queue-state becomes a symmetric 1-dimensional random walk with single boundary. It is well known, that such a process is only *null*-recurrent. Hence, $\bar{a} = 1$ (which is the boundary of $\mathcal{A}$) does not belong to $\mathcal{A}$.

One Policy that achieves throughput optimality is the already mentioned **MW policy**, given via

$$v_t = \min_{v_t\in\mathcal{V}} \; q_t^\intercal R\bar{M}_t v_t \qquad \text{s.t.} \qquad -R^- v_t \leq q_t \tag{2.6}$$

Essentially, MW uses the difference in queues states $(q_t^\intercal R)$, multiplied by the transmission success probability of the links between queues $(\bar{M}_t)$ to decide upon the control v_t, with larger differences and higher probabilities being favored. Its throughput optimality results from its ability to use misplaced packets as an indicator in subsequent time-steps (through $q_t^\intercal R$).

2.2.3 Predictive Network Control (PNC)

To control the introduced networks we propose a novel control policy that we refer to as Predictive Network Control (PNC). This policy operates under the paradigms of MPC and therefore is equipped with a modified copy of the system-model, called the *prediction-model*, and an *objective*. Its operation scheme is deterministic ($\omega_t \equiv 0$) and can be described as follows: (i) Based on the prediction-model, an entire trajectory of control-vectors from t up until $t + H - 1$ is optimized towards the objective, where H is called the prediction horizon. (ii) Only the first (i.e. the immediate) control-vector in this trajectory is actually applied to the system. (iii) The process repeats (discarding the rest of the just calculated trajectory).

W.l.o.g. we assume the current time-step to be 0 and that the policy has observed q_0 and s_0. Let σ_t be the probability distribution of s_t such that $\sigma_t^{[\mathfrak{s}]} = \mathbb{P}[s_t = \mathfrak{s} \mid \sigma_0]$ and $\sigma_t = \sigma_0 P^t$ (t being an exponent). Note that it makes no difference whether we condition an expression on s_0 or σ_0. For an arbitrary future time-step T, we express an expected queue-state in our system-model via

$$\mathbb{E}[q_T \mid q_0, \sigma_0] = q_0 + \sum_{t=0}^{T-1} \mathbb{E}\left[R W^{s_t} \phi(q_t, s_t, 0) \,\middle|\, q_0, \sigma_0 \right] + T\bar{a}. \tag{2.7}$$

To ease this prediction and thereby the computational effort, the prediction-model does not necessarily have to reflect the complexity of the system-model and can be designed in roughly three different ways (differing in the amount of future stochastically induced variations, that are considered):

(i) In a *true prediction*, the controller assigns a control-vector to *every* time-step (up until H) and *every* possible set of realizations of q_t, s_t along the way. The number of control-vectors required for such a prediction amounts roughly to $H \cdot n_s \cdot k$, where k is the number of all possible queue-states, that can (on average) be realized in a single time-step. This requires the maximum amount of computational resources but would truly find the optimal control trajectory that optimizes any given objective.

(ii) In contrast, a *relaxed prediction* uses only a minimum of control-vectors (though still enough to count as a meaningful prediction), i.e. for every time-step in $\{0, \dots H - 1\}$, only a single control-vector is utilized. This amounts to H control-vectors being required for the prediction, speeding up the calculation of an optimal control trajectory considerably. However, said trajectory might be sub-optimal compared to case i) and hence control performance might suffer.

(iii) Finally, a mixture of both cases could be implemented in order to balance computational complexity and control performance. E.g. for each time-step, one could condition the control-vector on the realization of s_t, but ignore realizations of q_t, leading to $H \cdot n_s$ control-vectors being used for the prediction.

Notably, we found that it suffices to define our prediction-model via case (ii) since we shall prove that already then throughput optimality is actually achieved. Let us denote the virtual queue-states and control-vectors in the prediction-model with $\dot{q}_t, \dot{v}_t$ (no connection to the time-derivative intended). Then an expected future queue-state *in the prediction-model* becomes

$$\mathbb{E}[\dot{q}_T \mid q_0, \sigma_0] = q_0 + \sum_{t=0}^{T-1} \left(\sum_{\mathfrak{s}\in\mathcal{S}} \sigma_t^{[\mathfrak{s}]} R W^{\mathfrak{s}} \right) \dot{v}_t + T\bar{a} \tag{2.8}$$

Because the prediction is initialized in each time-step again (using the recent measurements of $\{q_t\}$ and $\{s_t\}$, which are q_0 and s_0), (2.8) depends on q_0 and σ_0.

With that, the prediction-model is sufficiently defined and we are left with choosing an objective. To this end, the following abbreviations are useful:

$$\tilde{\dot{v}}_{0,T} := \begin{pmatrix} \dot{v}_0 \\ \vdots \\ \dot{v}_T \end{pmatrix}, \qquad \dot{\Delta}_{0,T} = \dot{\Delta}_{0,T}(\tilde{\dot{v}}_{0,T-1}) := \dot{q}_T - q_0 \qquad \|\dot{q}_t\|^2 := \dot{q}_t^\intercal Q \dot{q}_t. \tag{2.9}$$

Here, $\tilde{\dot{v}}_{0,T-1}$ is a *control trajectory*, $\dot{\Delta}_{0,T}$ is a difference between queue-states and the latter is an *affine* function of the former. The matrix Q is positive-semidefinite and could easily be carried through every derivation in this paper. However, for notational convenience we set $Q = I$ (the identity matrix).

A common quadratic objective is then formulated as

$$\mathbb{E}\left[\sum_{t=1}^{H} \dot{q}_t^\intercal Q \dot{q}_t \,\middle|\, q_0, s_0 \right] = \sum_{t=1}^{H} \mathbb{E}\left[\, \|\dot{q}_t\|^2 \mid q_0, s_0 \right] \tag{2.10}$$

It penalizes large queue-states in a quadratic fashion, and thereby motivates the control to deliver packets to their destinations. Looking at one of its representative terms

$$\|\dot{q}_t\|^2 = \|q_0\|^2 + \left\| \dot{\Delta}_{0,t} \right\|^2 + 2 q_0^\intercal \dot{\Delta}_{0,t} \tag{2.11}$$

we notice that $\dot{\Delta}_{0,t}$ is bounded (as proven later in Lemma 1) and q_0 is just an offset independent of $\tilde{\dot{v}}_{0,t-1}$. Hence, if q_0 is large enough and we are to minimize $\|\dot{q}_t\|^2$ over $\tilde{\dot{v}}_{0,t-1}$, the last term of (2.11) will always dominate the minimization. It is therefore sufficient to minimize over the terms $2q_0^\intercal \dot{\Delta}_{0,t}$, and hence *our* actual objective becomes

$$\begin{aligned} \mathbb{E}\left[\sum_{t=1}^{H} 2 q_0^\intercal \dot{\Delta}_{0,t} \,\middle|\, q_0, s_0 \right] &= 2q_0^\intercal \left(\sum_{t=0}^{H-1} \sum_{t}^{H} R \bar{W}_t(s_0) \dot{v}_t + \bar{a} \right) \\ &= l(q_0, s_0) \tilde{\dot{v}}_{0,H-1} + H(H+1) q_0^\intercal \bar{a} \end{aligned} \tag{2.12}$$

where $\bar{W}_t(s_0)$ and $l(q_0, s_0)$ are denoted in Table 2.1 and $\sum_t^H = H - t + 1$. This linear formulation has considerably reduced complexity compared to the initial quadratic one. Note that a similar step is taken in the derivation of the MW policy.

Finally, we must make sure that our trajectory fulfills the constraints not only for the immediate control-vector but for every control-vector in the entire trajectory (otherwise the predicted evolution of the system might never be achievable). This can be easily done by expanding the constraints in a block-diagonal fashion. Calculating the expanded constraint sets $\mathcal{B}, \mathcal{C}, \mathcal{R}(q_0)$ (for binary-, constituency-, routing-constraints) is a tedious but straightforward task, the results of which are denoted in Table 2.1. With that, the definition of the PNC policy with prediction horizon H can be given as

$$\begin{aligned}\phi^{\mathrm{PNC}}(q_0, s_0) &= \operatorname*{farg\,min}_{\tilde{v}_{0,H-1}} \mathbb{E}\left[\sum_{t=1}^{H} 2q_0^\intercal \dot{\Delta}_{0,t} \,\middle|\, q_0, s_0\right] \\ &= \operatorname*{farg\,min}_{\tilde{v}_{0,H-1}} l(q_0, s_0)\tilde{v}_{0,H-1} \qquad (2.13) \\ \text{subject to} \qquad & \tilde{v}_{0,H-1} \in \mathcal{B} \cap \mathcal{C} \cap \mathcal{R}(q_0)\end{aligned}$$

where farg min() expresses that only the first argument of the trajectory is used as output. The optimization can be classified as a binary linear program whose feasible set always contains $\tilde{v}_{0,H-1} = \mathbf{0}$. Hence, a not necessarily unique optimal solution can always be obtained via conventional branch and bound methods. It is readily verified that $\{q_t, s_t\}$ becomes a homogeneous DTMC under the PNC policy (meaning that its transition matrix is independent of t).

Remark: Choosing $H = 1$ yields the common MW policy. Indeed, if the PNC controller would follow a once calculated optimal trajectory to its end (i.e. for H time-steps), PNC would merely be the extension of MW over multiple time-steps. However, PNC recalculates the trajectory each time-step, thereby discarding its entire tail. This results in a much improved behavior of PNC (see Section 2.2.5) but also makes it impossible to infer any properties from MW to PNC. For more comparisons between the two policies, we refer to [25] and [26].

2.2.4 Throughput Optimality of the PNC Policy

Following theorem is an integral contribution of this paper:

Theorem 1. *The PNC policy* (2.13) *is throughput optimal for the system* (2.2).

For the proof, we use the well-known theorem for positive-recurrent DTMCs by Foster [24]: Finding a Lyapunov-function over the state-space on which the system evolution "drifts downward" almost everywhere, guarantees positive-recurrence (stability). The difficulty in our case arises from the fact that the PNC policy (like any other MPC) discards all but the first control-vector of the optimal trajectory. Hence $v_0 = \phi^{\mathrm{PNC}}(q_0, s_0)$ is influenced by the expected evolution for the next H time-steps, yet the drift is necessarily defined over a one-step evolution. To resolve this mismatch, we employ a Lyapunov-function which is defined implicitly over a minimization that mimics (2.13). However, due to the need of being sensitive to topology-state s_1, this minimization is performed over a different model than the one employed in the PNC policy. Indeed, the required model for the prediction inside this Lyapunov-function is very much alike the one that was discussed under point (iii) in Section 2.2.3. Interfacing our PNC policy (which uses a much simpler model in order to maintain lowest possible complexity) is then achieved by first lifting the constraints, allowing us to manipulate the minimization and insert substitute variables, and then reintroducing the constraints to reintegrate the entire derivation into Foster's theorem.

Expected value of W^{s_t}	$\bar{W}_t(s_0) := \mathbb{E}[W^{s_t} \mid s_0] = \sum_{\mathfrak{s}\in\mathcal{S}} \sigma_t^{[\mathfrak{s}]} W^{\mathfrak{s}}$	(2.14)
Linear objective vector $l(q_0, s_0)$	$l(q_0,s_0) = 2q_0^\intercal R \begin{pmatrix} (H-0)\bar{W}_0^\intercal(s_0) \\ (H-1)\bar{W}_1^\intercal(s_0) \\ \vdots \\ \bar{W}_{H-1}^\intercal(s_0) \end{pmatrix}^\intercal$	(2.15)
Binary-constraints	$\tilde{x}_{1,H} \in \mathcal{B} \iff \tilde{x}_{1,H} \in \{0,1\}^{n_v \cdot H \cdot D}$	(2.16)
Constituency-constraints	$\tilde{x}_{1,H} \in \mathcal{C} \iff (I_D \otimes I_H \otimes C)\,\tilde{x}_{1,H} \leq \mathbf{1}_D \otimes \mathbf{1}_H \otimes c$	(2.17)
Routing-constraints	$\tilde{x}_{1,H} \in \mathcal{R}(q_0) \iff$ $\begin{pmatrix} \mathbf{1}_D \otimes R^- & & & \\ \mathbf{1}_D \otimes R & \mathbf{1}_D \otimes R^- & & \\ \vdots & & \ddots & \\ \mathbf{1}_D \otimes R & \dots & \mathbf{1}_D \otimes R & \mathbf{1}_D \otimes R^- \end{pmatrix} \tilde{x}_{1,H} \leq \begin{pmatrix} \mathbf{1}_D \otimes q_0 \\ \mathbf{1}_D \otimes (q_0 + \bar{a}) \\ \vdots \\ \mathbf{1}_D \otimes (q_0 + (H-1)\bar{a}) \end{pmatrix}$	(2.18)
$D = 1$ for the prediction-model $D = n_s$ for the proof-model $\sigma_t = \sigma_0 P^t$ (t is an exponent) I_D is the identity matrix with dimension D		

Table 2.1: Quantities for policy and proof.

Preliminaries

It will often become necessary to upper and lower bound certain expressions. For this, we will use $k^1, k^2, k^3, \dots$ to denote arbitrary, non-zero, positive constants which are *independent* of the initial queue-state q_0 and whose exact values are of no further interest. Gothic letters are used to express realizations of random variables, such that e.g. $\mathfrak{s}_t$ is a realization of s_t, hence $\mathfrak{s}_t \in \mathcal{S}$. At some instances we use the trivial summation $\sum_x^y = y - x + 1$ in order to maintain well-arranged equations. The zero-vector is $\mathbf{0}_n$ and the vector of ones is $\mathbf{1}_n$, where the index stands for the dimension. If the argument of a minimization is not denoted, it is the same as the argument of the prior minimization in the derivation. Lastly, remember that a control-vector is always binary. We will therefore neglect denoting this constraint ($x \in \mathcal{B}$), which simplifies the formulas. With that we formulate the next lemmas, needed for the proof.

Lemma 1. *The difference $\Delta_{0,T}$ (or $\dot{\Delta}_{0,T}$ respectively) between two queue-states from time-steps* 0 *and* T *is bounded (element-wise) by*

$$-Tn_v\mathbf{1}_{n_q} \leq \Delta_{0,T} \leq T(n_v + \hat{a})\mathbf{1}_{n_q} \tag{2.19}$$

leading to

$$\|q_0\|^2 + 2q_0^\intercal\Delta_{0,T} \leq \|q_T\|^2 \leq \|q_0\|^2 + 2q_0^\intercal\Delta_{0,T} + k^1 \tag{2.20}$$

Simply put: knowing q_0 we can bound $\|q_T\|^2$.

Proof. Between time-steps 0 and T we have at best a constant efflux of n_v or at worst a constant influx of $n_v + \hat{a}$ packets per queue per time-step (since there are at most n_v links to fill or drain any given queue). □

Lemma 2. *The difference between the minimization defined by* (2.13)*, and the same minimization but relaxed by ignoring the routing-constraints can be bounded by*

$$\begin{aligned} &\min_{\tilde{v}_{0,H-1}\in\mathcal{C},\mathcal{R}(q_0)} \mathbb{E}\left[\sum_{t=1}^{H} 2q_0^\intercal\dot{\Delta}_{0,t} \,\middle|\, q_0, s_0\right] \\ -&\min_{\tilde{v}_{0,H-1}\in\mathcal{C}} \mathbb{E}\left[\sum_{t=1}^{H} 2q_0^\intercal\dot{\Delta}_{0,t} \,\middle|\, q_0, s_0\right] \leq n_v n_q H\,(H+1)\,(Hn_v - 1) = k^2 \end{aligned} \tag{2.21}$$

Simply put: the effect of the queue-state dependent routing-constraints $\mathcal{R}(q_0)$ on the solution of the minimization can be bounded.

Proof. First, consider $\dot{\Delta}_{0,t}$: the maximum difference that $\dot{\Delta}_{0,t}$ can generate is less than $tn_v\mathbf{1}_{n_q}$ (all links drain a queue over t steps). Second, consider q_0. Again, for a single queue, the *most* efflux in H time-steps is Hn_v packets (if every link would drain that queue). Hence, if $q_0 \geq Hn_v\mathbf{1}_{n_q}$, a control trajectory *cannot* violate the routing-constraints. Conversely, if a link cannot be activated due to the routing-constraints, at least one entry of q_0 must be smaller than Hn_v.

Due to the linearity, the largest difference in the minimizations will be found when $q_0 = (Hn_v - 1)\mathbf{1}_{n_q}$ (possibly denying any activation under routing-constraints). This, together with the initial bound on $\dot{\Delta}_{0,t}$ leads directly to

$$\sum_{t=1}^{H} 2(Hn_v - 1)\mathbf{1}_{n_q}^\intercal tn_v\mathbf{1}_{n_q} = n_v n_q H\,(H+1)\,(Hn_v - 1) \tag{2.22}$$

which is an upper bound for the difference in question. □

Finally, the following theorem will allow us to express our definition of stability by the means of a Lyapunov-function.

Lemma 3. *If $\{q_t, s_t\}$ is a homogeneous DTMC, $\{q_t, s_t\}$ is positive-recurrent if there exists a function $f : \mathcal{Q} \times \mathcal{S} \to \mathbb{R}_+$ such that $\forall q_0 \in \mathcal{Q}$:*

$$\mathbb{E}[f(q_1, s_1) - f(q_0, s_0) \,|\, q_0] \leq k^4 - k^5\mathbf{1}_{n_q}^\intercal q_0 \tag{2.23}$$

Simply put: if expected difference between two consecutively taken values on f is negative almost everywhere, the DTMC must be positive-recurrent.

Proof. This is a simple application of the recurrence theorem for DTMCs by Foster [24] considering that $\{s_t\}$ is itself a finite and therefore positive-recurrent DTMC. In particular, one can follow the proof from [27, p. 168] and note that because $\{s_t\}$ is finite, the return time to the finite set is independent of $\{s_t\}$. Beyond that it is easy to verify that f fulfills the conditions on the employed function. □

Main Proof of Theorem 1

We now start with the main part of the proof. We will define a Lyapunov-function $f(q_t, s_t)$ and show that *if* the system is governed by the PNC policy with horizon $H + 1$, f fulfills Lemma 3 for any $\bar{a} \in \mathcal{A}$.

In addition to the system- and prediction-model we introduce the proof-model (named this way because we only need it in this proof). It mimics the system-model except that the control-vectors are only a function of s_t (and, of course, t). We will denote virtual queue-state and control-vector in the proof-model as $\ddot{q}_t, \ddot{v}_t(s_t)$ (no connection to the time-derivative intended). A prediction of future queue-states in the proof-model results in

$$\mathbb{E}[\ddot{q}_T \mid q_0, \sigma_0] = q_0 + \sum_{t=0}^{T-1} \sum_{\mathfrak{s} \in \mathcal{S}} \sigma_t^{[\mathfrak{s}]} R W^{\mathfrak{s}} \ddot{v}_t(\mathfrak{s}) + T\bar{a} \tag{2.24}$$

Once again, q_0 and σ_0 are quantities from the system-model that are used as a fix point from which the prediction is started. We define a control trajectory for the proof-model as

$$\tilde{\ddot{v}}_{0,H} = \begin{pmatrix} \ddot{v}'_0 \\ \ddot{v}'_1 \\ \vdots \\ \ddot{v}'_H \end{pmatrix}, \qquad \text{with} \qquad \ddot{v}'_t = \begin{pmatrix} \ddot{v}_t(s_t = 1) \\ \ddot{v}_t(s_t = 2) \\ \vdots \\ \ddot{v}_t(s_t = n_s) \end{pmatrix} \tag{2.25}$$

and similar to (2.9) the difference of queue-states as $\ddot{\Delta}_{0,T} = \ddot{q}_T - q_0$. Note that Lemma 1 holds equally for $\Delta, \dot{\Delta}$ and $\ddot{\Delta}$.

For a PNC policy with horizon $H + 1$, we employ the following Lyapunov-function:

$$f(q_0, s_0) = \min_{\tilde{\ddot{v}}_{0,H-1} \in \mathcal{C}} \mathbb{E}\left[\sum_{t=1}^{H} \|\ddot{q}_t\|^2 \,\middle|\, q_0, s_0 \right] \tag{2.26}$$

Note that the horizon of the minimization in f is chosen one step smaller than that of the PNC policy and the control trajectory is *not* subject to routing-constraints $\mathcal{R}(q_0)$.

We can now start expressing the first term of (2.23) (for now also conditioning on s_0) like

$$\begin{aligned}
\mathbb{E}[f(q_1,s_1)\,|\,q_0,s_0] &= \mathbb{E}\left[\min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}} \mathbb{E}\left[\sum_{t=2}^{H+1}\|\ddot{q}_t\|^2 \,\middle|\, q_1,s_1\right] \,\middle|\, q_0,s_0\right] \\
&\le \min \mathbb{E}\left[\mathbb{E}\left[\sum_{t=2}^{H+1}\|\ddot{q}_t\|^2 \,\middle|\, q_1,s_1\right] \,\middle|\, q_0,s_0\right] \\
&\le \min \mathbb{E}\left[H\|q_1\|^2 + k^6 + \mathbb{E}\left[\sum_{t=2}^{H+1} 2q_1^\intercal\ddot{\Delta}_{1,t} \,\middle|\, q_1,s_1\right] \,\middle|\, q_0,s_0\right] \\
&\le \min \mathbb{E}\left[H\|q_0\|^2 + H2q_0^\intercal\Delta_{0,1} + k^7 + \sum_{t=2}^{H+1} 2(q_0+\Delta_{0,1})^\intercal\ddot{\Delta}_{1,t} \,\middle|\, q_0,s_0\right] \\
&\le H\|q_0\|^2 + k^8 + \min 2q_0^\intercal\mathbb{E}\left[\sum_{t=2}^{H+1}\ddot{\Delta}_{1,t} + H\Delta_{0,1} \,\middle|\, q_0,s_0\right] \qquad (2.27) \\
&\le H\|q_0\|^2 + k^8 + \sum_{t=0}^{H}\sum_{t}^{H} 2q_0^\intercal\bar{a} - \min_{v_0\in\mathcal{C}}\mathbb{E}[2q_0^\intercal\Delta_{0,1}\,|\,q_0,s_0] \\
&\quad + \min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}} 2q_0^\intercal R\left(\sum_{0}^{H} W^{s_0}v_0 + \sum_{t=1}^{H}\sum_{t}^{H}\sum_{\mathfrak{s}\in\mathcal{S}}\sigma_t^{[\mathfrak{s}]}W^{\mathfrak{s}}\ddot{v}_t(\mathfrak{s})\right) \qquad (2.28)
\end{aligned}$$

First and second inequality follow from expressing the expectation as a sum and applying Lemma 1, respectively. To reach (2.27) we substitute $q_1 = q_0 + \Delta_{0,1}$ and bound again with Lemma 1. Note that the product $\Delta^\intercal\ddot{\Delta}$ can also be bounded accordingly. Finally, (2.28) follows by writing out any Δ term, using (2.24), and both adding and subtracting an additional $\Delta_{0,1}$ term. The latter is also subject to minimization which makes (2.28) become an inequality.

Next, we concentrate on the last minimization of (2.28). Our goal will be to extend the argument from $\tilde{\ddot{v}}_{1,H}$ to $\tilde{\ddot{v}}_{0,H}$ using the PNC policy. To that end we make a change in variables: Instead of using $\ddot{v}_t(s_t = 1), \ldots \ddot{v}_t(s_t = n_s)$, we use a *common* vector $\mu_t \in \{0,1\}^{n_v}$, and a set of discrepancies $\delta_t(s_t = 1), \ldots \delta_t(s_t = n_s)$ with $\delta_t(s_t) \in \{-1,0,+1\}^{n_v}$ and $\mu_t + \delta_t(s_t) \in \{0,1\}^{n_v}$. We decompose the control trajectory according to

$$\ddot{v}'_t = \begin{pmatrix}\ddot{v}_t(1)\\ \vdots \\ \ddot{v}_t(n_s)\end{pmatrix} = \begin{pmatrix}\mu_t\\ \vdots\\ \mu_t\end{pmatrix} + \begin{pmatrix}\delta_t(1)\\ \vdots\\ \delta_t(n_s)\end{pmatrix} = \mu'_t + \delta'_t, \qquad (2.29)$$

$$\tilde{\ddot{v}}_{1,H} = \begin{pmatrix}\ddot{v}'_1\\ \vdots\\ \ddot{v}'_H\end{pmatrix} = \begin{pmatrix}\mu'_1\\ \vdots\\ \mu'_H\end{pmatrix} + \begin{pmatrix}\delta'_1\\ \vdots\\ \delta'_H\end{pmatrix} = \tilde{\mu}_{1,H} + \tilde{\delta}_{1,H}$$

Crucially, we enforce the constituency-constraints $\tilde{\mu}_{1,H} \in \mathcal{C}$. But once $\tilde{\mu}_{1,H}$ is fixed, we still need to abide to the constituency of $\tilde{\ddot{v}}_{1,H}$. We will denote this as $(\tilde{\delta}_{1,H} \in \mathcal{C}_\mu) := (\tilde{\mu}_{1,H} + \tilde{\delta}_{1,H} \in \mathcal{C})$. This entire transformation is necessary in order to bring v_0 (selected by PNC) into the minimization operator. The PNC policy is defined as an optimization over multiple control-vectors and we need to identify this optimization somewhere in the Lyapunov function. The transformation will help us with this identification since the trajectory $\tilde{\mu}_{1,H}$ has the same impact on the system as $\tilde{\ddot{v}}_{1,H}$, which can be seen in the next derivation.

With $\tilde{\mu}_{1,H}$ and $\tilde{\delta}_{1,H}$ the last term of (2.28) becomes

$$
\begin{aligned}
&\min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}} 2q_0^{\mathsf{T}}R\left(\sum_0^H W^{s_0}v_0 + \sum_{t=1}^H\sum_t^H\sum_{\mathfrak{s}\in\mathcal{S}}\sigma_t^{[\mathfrak{s}]}W^{\mathfrak{s}}\ddot{v}_t(\mathfrak{s})\right)\\
&= \min_{\tilde{\mu}_{1,H}+\tilde{\delta}_{1,H}\in\mathcal{C}} 2q_0^{\mathsf{T}}R\left(\sum_0^H W^{s_0}v_0 + \sum_{t=1}^H\sum_t^H\sum_{\mathfrak{s}\in\mathcal{S}}\sigma_t^{[\mathfrak{s}]}W^{\mathfrak{s}}\left(\mu_t+\delta_t(\mathfrak{s})\right)\right)\\
&\overset{(2.14)}{=} \min_{\tilde{\delta}_{1,H}\in\mathcal{C}_\mu}\ \min_{\tilde{\mu}_{1,H}\in\mathcal{C}} 2q_0^{\mathsf{T}}R\left(\sum_0^H W^{s_0}v_0 + \sum_{t=1}^H\sum_t^H \bar{W}_t(s_0)\mu_t\right.\\
&\qquad\left. + \sum_{t=1}^H\sum_t^H\sum_{\mathfrak{s}\in\mathcal{S}}\sigma_t^{[\mathfrak{s}]}W^{\mathfrak{s}}\delta_t(\mathfrak{s})\right) &(2.30)\\
&\le \min_{\tilde{\delta}_{1,H}\in\mathcal{C}_\mu}\ \min_{\left(\mathbf{0}^{\mathsf{T}},(\tilde{\mu}_{1,H})^{\mathsf{T}}\right)^{\mathsf{T}}\in\mathcal{C}\cap\mathcal{R}(q_0)} 2q_0^{\mathsf{T}}R(\dots) &(2.31)\\
&\overset{\text{PNC}}{=} \min_{\tilde{\delta}_{1,H}\in\mathcal{C}_\mu}\ \min_{\left(v_0^{\mathsf{T}},(\tilde{\mu}_{1,H})^{\mathsf{T}}\right)^{\mathsf{T}}\in\mathcal{C}\cap\mathcal{R}(q_0)} 2q_0^{\mathsf{T}}R(\dots) &(2.32)\\
&\overset{\text{Lemma 2}}{\le} k^9 + \min_{\tilde{\delta}_{1,H}\in\mathcal{C}_\mu}\ \min_{\left(v_0^{\mathsf{T}},(\tilde{\mu}_{1,H})^{\mathsf{T}}\right)^{\mathsf{T}}\in\mathcal{C}} 2q_0^{\mathsf{T}}R(\dots)\\
&\le k^9 + \min_{\tilde{\delta}_{1,H}\in\mathcal{C}_\mu}\ \min_{v_0\in\mathcal{C}}\ \min_{\tilde{\mu}_{1,H}\in\mathcal{C}} 2q_0^{\mathsf{T}}R(\dots) &(2.33)\\
&= k^9 + \min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}}\ \min_{v_0\in\mathcal{C}} 2q_0^{\mathsf{T}}R(\dots) &(2.34)\\
&= k^9 + \min_{\tilde{\ddot{v}}_{0,H}\in\mathcal{C}} 2q_0^{\mathsf{T}}R\left(\sum_{t=0}^H\sum_t^H\sum_{\mathfrak{s}\in\mathcal{S}}\sigma_t^{[\mathfrak{s}]}W^{\mathfrak{s}}\ddot{v}_t(\mathfrak{s})\right) &(2.35)\\
&= k^9 + \min_{\tilde{\ddot{v}}_{0,H}\in\mathcal{C}} \mathbb{E}\left[\sum_{t=1}^{H+1} 2q_0^{\mathsf{T}}\ddot{\Delta}_{0,t}\,\middle|\, q_0, s_0\right] - \sum_{t=0}^H\sum_t^H 2q_0^{\mathsf{T}}\bar{a} &(2.36)
\end{aligned}
$$

We are allowed to separate the minimizations in (2.30) because this does not increase the overall feasible set and hence still yields the same value for $2q_0^{\mathsf{T}}R(\dots)$. Inequality (2.31) results from introducing the routing-constraints $\mathcal{R}(q_0)$ which decreases the feasible set for $\tilde{\mu}_{1,H}$. Since $\mathcal{R}(q_0)$ depends on q_0, the zero-vector of dimension $n_s\cdot n_v$ has to be attached to the trajectory $\tilde{\mu}_{1,H}$ in order to extend the trajectory back to time-step 0. Now, because $W^{s_0}=\bar{W}_0(s_0)$ (taken from (2.30) and (2.12), respectively) we can identify the PNC policy in the derivation: The choice of v_0 together with the minimization over $\left(\mathbf{0}^{\mathsf{T}},(\tilde{\mu}_{1,H})^{\mathsf{T}}\right)^{\mathsf{T}}$ results in the same value for $2q_0^{\mathsf{T}}R(\dots)$ as the minimization over the trajectory $\left(v_0^{\mathsf{T}},(\tilde{\mu}_{1,H})^{\mathsf{T}}\right)^{\mathsf{T}}$, but *only because* v_0 is governed by the PNC policy. In other words, $\left(v_0^{\mathsf{T}},(\tilde{\mu}_{1,H})^{\mathsf{T}}\right)^{\mathsf{T}}$ behaves exactly like $\tilde{\ddot{v}}_{0,H}$. From there, Lemma 2 allows us to drop the routing-constraints $\mathcal{R}(q_0)$ which in turn allows to separate the minimizations due to independence in (2.33). We take back the change in variables in (2.34) and merge v_0 and $\tilde{\ddot{v}}_{1,H}$ to $\tilde{\ddot{v}}_{0,H}$ in (2.35). This merging is possible because only the s_0-th control-vector in $\ddot{v}_0'$ (from $\tilde{\ddot{v}}_{0,H}$, see (2.29)) will have impact on $2q_0^{\mathsf{T}}R(\dots)$. The other control-vectors in $\ddot{v}_0'$ are instead multiplied with the 0 entries of σ_0. Hence, minimizing over $\ddot{v}_0'$ or v_0 yields the same result.

Combining (2.28) and (2.36) we are left with

$$\begin{aligned}&\mathbb{E}[f(q_1,s_1)\,|\,q_0,s_0]-H\,\|q_0\|^2-k^{10}\\ \le\;&\min_{\tilde{\ddot{v}}_{0,H}\in\mathcal{C}}\mathbb{E}\left[\sum_{t=1}^{H+1}2q_0^\intercal\ddot{\Delta}_{0,t}\,\middle|\,q_0,s_0\right]-\min_{\ddot{v}_0'\in\mathcal{C}}\mathbb{E}\left[2q_0^\intercal\ddot{\Delta}_{0,1}\,\middle|\,q_0,s_0\right]\end{aligned}\tag{2.37}$$

Again, we exchanged $\Delta_{0,1}$ with $\ddot{\Delta}_{0,1}$ and a corresponding control because s_0 is known. Hence, in both cases, only a single control-vector of the argument actually influences the objective, and in both cases the effect is the same. This completes the derivation for the first term of (2.23).

The second term of (2.23) has no interaction with the PNC policy and therefore can be easily obtained, in the same fashion as (2.28):

$$\mathbb{E}[f(q_0,s_0)\,|\,q_0,s_0]-H\,\|q_0\|^2\ge\min_{\tilde{\ddot{v}}_{0,H-1}\in\mathcal{C}}\mathbb{E}\left[\sum_{t=1}^{H}2q_0^\intercal\ddot{\Delta}_{0,t}\,\middle|\,q_0,s_0\right]\tag{2.38}$$

Combining (2.37) and (2.38) results in

$$\mathbb{E}[f(q_1,s_1)-f(q_0,s_0)\,|\,q_0,s_0]\le k^{10}+\min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}}\mathbb{E}\left[2q_0^\intercal\ddot{\Delta}_{1,H+1}\,\middle|\,q_0,s_0\right]\tag{2.39}$$

To alleviate the conditioning on s_0, we take the expectation $\mathbb{E}[\cdot\,|\,q_0]$ on both sides. After that we can swap minimization and expectation operator and write out the $\ddot{\Delta}$ terms:

$$\begin{aligned}\mathbb{E}[f(q_1,s_1)-f(q_0,s_0)\,|\,q_0]&\le k^{10}+\mathbb{E}\left[\min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}}\mathbb{E}\left[2q_0^\intercal\ddot{\Delta}_{1,H+1}\,\middle|\,q_0,s_0\right]\,\middle|\,q_0\right]\\&\le k^{10}+\min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}}\mathbb{E}\left[2q_0^\intercal\ddot{\Delta}_{1,H+1}\,\middle|\,q_0\right]\\&=k^{10}+\min_{\tilde{\ddot{v}}_{1,H}\in\mathcal{C}}2q_0^\intercal\left(\sum_{t=1}^{H}\sum_{\mathfrak{s}\in\mathcal{S}}\pi^{[\mathfrak{s}]}RW^{\mathfrak{s}}\ddot{v}_t(\mathfrak{s})+H\bar{a}\right)\\&=k^{10}+\min_{\ddot{v}_0'\in\mathcal{C}}2Hq_0^\intercal\left(\sum_{\mathfrak{s}\in\mathcal{S}}\pi^{[\mathfrak{s}]}RW^{\mathfrak{s}}\ddot{v}_0(\mathfrak{s})+\bar{a}\right)\end{aligned}\tag{2.40}$$

The last equality follows from the fact that each $\ddot{v}_t'$ from $\tilde{\ddot{v}}_{1,H}$ encounters the same term in the objective. Hence, the minimization can be formulated over a single vector from the proof-model, which is chosen to be $\ddot{v}_0'$. Finally, we are interested in those cases for which $\bar{a}\in\mathcal{A}$ (and hence stabilization is achievable). From (2.5) we know that in those cases $\bar{a}=-\sum_{\mathfrak{s}\in\mathcal{S}}\pi^{[\mathfrak{s}]}RW^{\mathfrak{s}}\bar{v}^{\mathfrak{s}}-k^{11}\mathbf{1}_{n_q}$ where $\bar{v}^{\mathfrak{s}}$ was an element from $\operatorname{conv}(\mathcal{V})$ and $k^{11}>0$ is some small constant. It follows that

$$\begin{aligned}&\mathbb{E}[f(q_1,s_1)-f(q_0,s_0)\,|\,q_0]\\&\le k^{10}+\min_{\ddot{v}_0'\in\mathcal{C}}2Hq_0^\intercal\left(\sum_{\mathfrak{s}\in\mathcal{S}}\pi^{[\mathfrak{s}]}RW^{\mathfrak{s}}\ddot{v}_0(\mathfrak{s})-\sum_{\mathfrak{s}\in\mathcal{S}}\pi^{[\mathfrak{s}]}RW^{\mathfrak{s}}\bar{v}^{\mathfrak{s}}-k^{11}\mathbf{1}_{n_q}\right)\end{aligned}\tag{2.41}$$

$$\le k^{10}+2Hq_0^\intercal\left(-\mathbf{1}_{n_q}k^{11}\right)=k^{10}-k^{12}\mathbf{1}_{n_q}^\intercal q_0\tag{2.42}$$

The constraint $\ddot{v}'_0 \in \mathcal{C}$ implies $\ddot{v}_0(\mathfrak{s}) \in \mathcal{V}$ for all $\mathfrak{s} \in \mathcal{S}$. Thus, $\ddot{v}'_0$ can lie on any edge of the boundary of $\operatorname{conv}(\mathcal{V})$. Since the minimization is linear it will find its optimum on one of those edges. Hence, we could replace $\ddot{v}_0(\mathfrak{s}) \in \mathcal{V}$ with $\ddot{v}_0(\mathfrak{s}) \in \operatorname{conv}(\mathcal{V})$ and not change the outcome of the minimization. But because $\bar{v}^{\mathfrak{s}}$ is also from $\operatorname{conv}(\mathcal{V})$, it follows that the first sum in (2.41) is guaranteed to be smaller than the second sum which leads to (2.42). This in turn fits to Lemma 3 and therefore proves positive-recurrence of $\{q_t, s_t\}$. Since this result was generated by applying the PNC policy and holds for all $\bar{a} \in \mathcal{A}$, throughput optimality follows. □

2.2.5 Exemplary Applications of PNC

This section holds simulation results which illustrate the performance of PNC compared to selected policies. We employ the following policies:

1. PNC, as introduced in Section 2.2.3
2. qPNC, equal to PNC but considers the entire quadratic term from (2.10) in the optimization (whereas PNC only considers the reduced linear term)
3. fPNC, equal to PNC but only calculates the optimal trajectory every H steps, otherwise follows the afore calculated trajectory
4. MW, as introduced in (2.6), equal to PNC with $H = 1$, represents the prototype back-pressure policy
5. RR, round robin policy operating in cycles; in each cycle, each link is activated once as part of a random order

For the simulations, we implemented the systems according to (2.2) and the PNC policy according to (2.13) and especially Table 2.1.

Dynamic Topology

We employ a scenario as depicted in Figure 2.2, where a mobile user equipment (UE) crosses multiple sectors, each one designated to a specific access point (AP). In each sector, the UE can only communicate with the corresponding AP. The APs are connected to a global network from which they receive packets that they are supposed to transmit to the UE. The derived queueing network is shown in Figure 2.3. We assume that the UE travels with constant velocity along a known path and the sectors do not overlap. The UE remains in each sector for exactly 3 time-steps, where it experiences perfect channel quality (guaranteed transmission success). A single packet is created every second time-step at $q^{[1]}$, which represents the entire arrival to the system. The controller can only select a single link per time-step.

Given the simplicity of the scenario, the DTMC $\{s_t\}$ becomes deterministic, allowing us to fix the time behavior of the parameters $\bar{m}_t^{[j]}$ (transmission success probabilities) of the links. In particular, each sequence $\{\bar{m}_t^{[j]}\}$ becomes a *binary* sequence. If j is uneven, $\bar{m}_t^{[j]} = 1$ (corresponding to the wired global network). Else $\bar{m}_t^{[j]} = 1$ only if the UE passes through the corresponding sector as depicted in Figure 2.4.

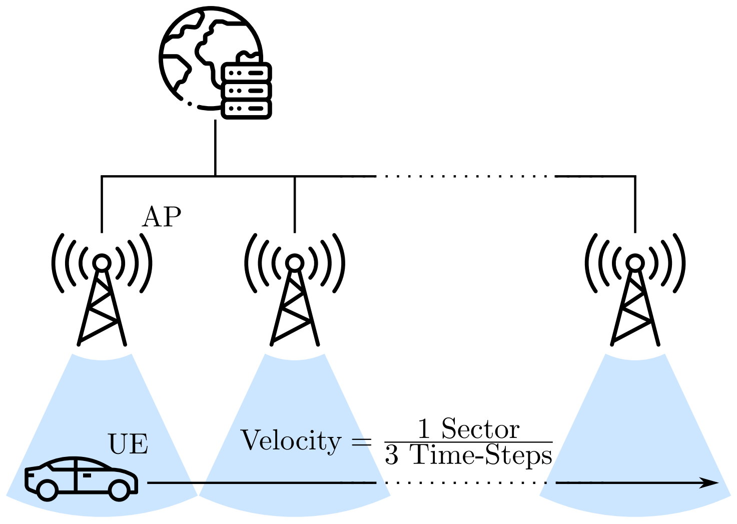


Fig. 2.2: Scenario for example 1. A user equipment traverses multiple access points with constant velocity, thereby being connected to various access points.

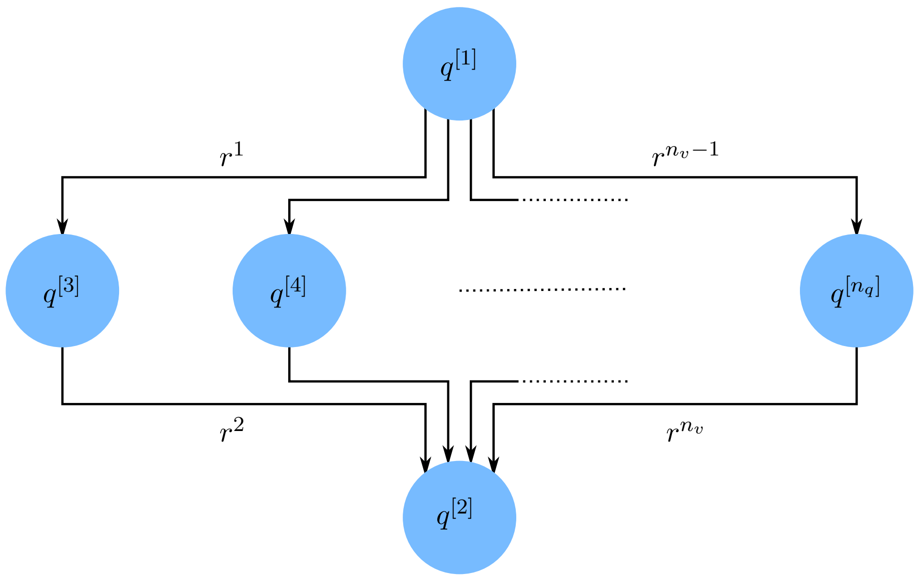


Fig. 2.3: Corresponding queueing network to Figure 2.2. Each node represents an access point, except $q^{[1]}$ and $q^{[2]}$ which represent global source and destination, respectively.

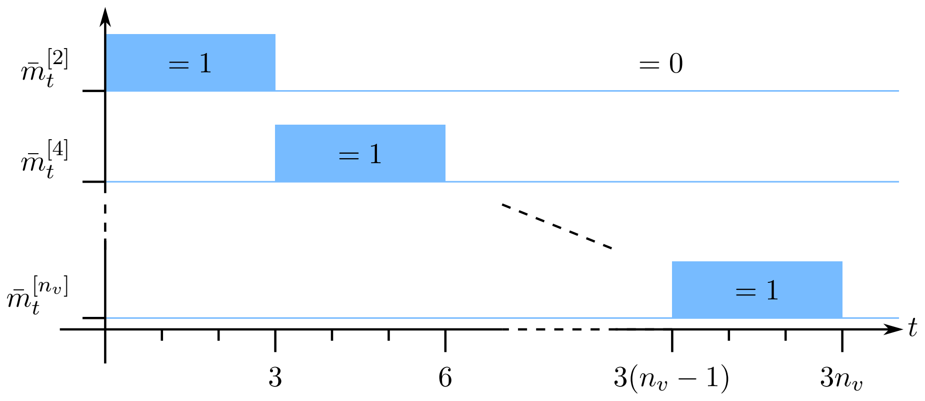


Fig. 2.4: Transmission success probabilities for the corresponding links to Figure 2.3. Links are either ideal (error-free) or cannot establish communication at all.

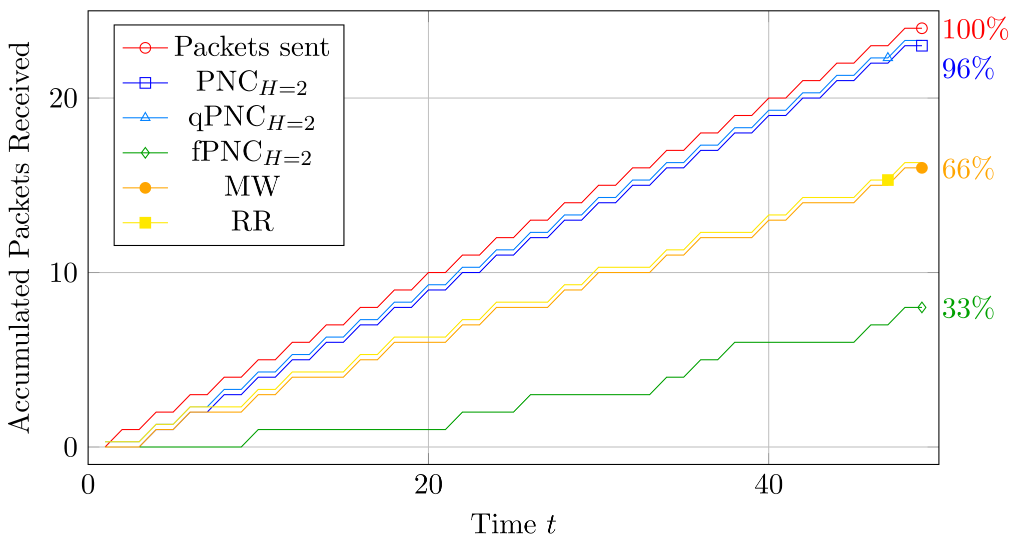


Fig. 2.5: Simulated throughput of packets, refering to Figure 2.2. Given a constant flow of packets (red) that is to be transmitted to the destination, our predictive policies are able to successfully deliver nearly all packets (blue). The MW policy does only manage do deliver 66% (orange).

The simulation results in Figure 2.5 show the accumulated amount of packets sent and received by the UE. Some plots are slightly shifted vertically to improve visualization. It can be observed that only around 66% of the packets reach the UE for the MW policy. The same holds for the RR policy which acts identically to MW for this scenario. In comparison, PNC and qPNC can deliver close to 100% of the packets, and this already when applying merely $H = 2$. This is quite obviously facilitated by the topology: It only takes 2 hops from the source of the packet to the UE and thus $H = 2$ is already sufficient to determine an optimal path through the network. Interestingly, fPNC performs much worse. This is because in some time-steps, fPNC still follows an old control trajectory which has become sub-optimal due to arrivals and changes in topology.

Networks with Assembly-Queues

Though MW performs poorly in networks with dynamic topology, as showcased in the previous simulation, it is still able to achieve throughput optimality in the long run if we assume MW to be sensitive to the current topology-state s_t (an assumption that we also used for PNC). Indeed, for conventional networks, throughput optimality is shared by both policies. However, in the example at hand we extend conventional networks with *assembly*-queues [28, 29, 30, 31, 32, 33, 34, 35]. In a batch of assembly-queues, all queues must be served at the same time. Conversely, if a single queue of such a batch is depleted, all other queues of the same batch cannot be served (at least not over that particular server or link). As the name suggests, those queues can not only be applied to assembly lines in industrial settings, but also help to model logistic and traffic scenarios, e.g. when parcels for the same customer are to be delivered together. In networks with assembly-queues, only PNC seems to maintain its throughput optimality while MW fails [25].

For our simulation, we imagine a scenario as depicted in Figure 2.6. Set-up and queueing network are shown on the left and right side, respectively. The example consists of an access point (AP) $q^{[1]}$ that can either transmit solitary (link r^1), or initiate synchronized transmission (link r^3) with a neighboring AP $q^{[2]}$. The synchronized transmission uses constructive interference and thus achieves higher success probability. However, before synchronized transmission can be initiated, the data packets have to be shared (link r^2), i.e. copied from $q^{[1]}$ to $q^{[2]}$. Only one link may be activated in each time-step.

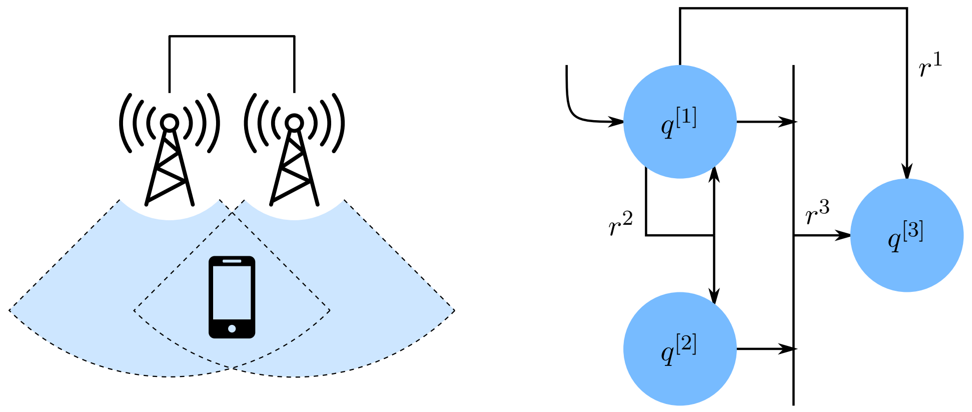
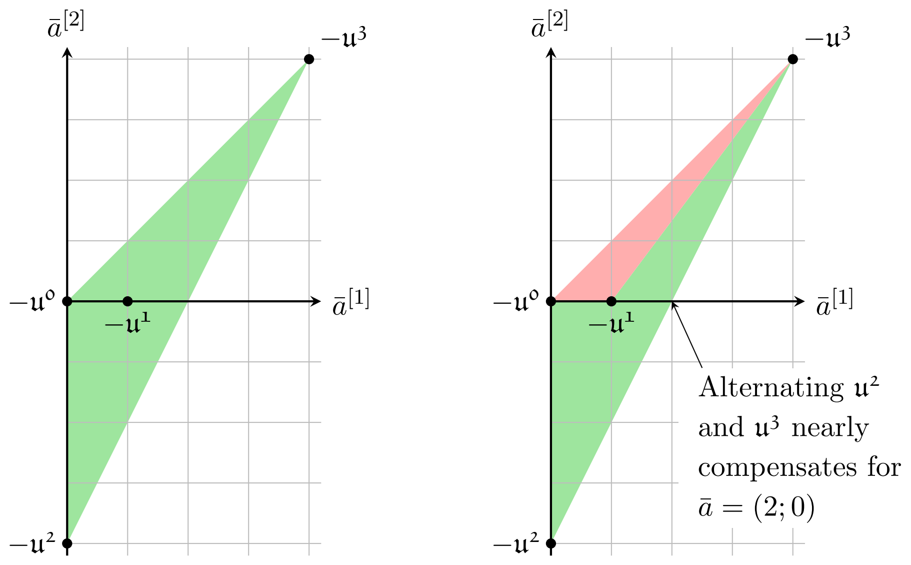


Fig. 2.6: Scenario and corresponding queueing network. One access point ($q^{[1]}$) can decide between direct transmission or synchronized transmission using a neighboring AP ($q^{[2]}$). For this to be relevant, transmission via r^3 must feature a much higher success probability compared to r^1.



Stability Region $\mathcal{A}$ (green)

Set of Arrival Rates $\bar{a}$ that MW is able to stabilize (red)

Fig. 2.7: Stability region for the queueing network depicted in Figure 2.6. MW is only able to stabilize a small part of the entire stability region.

In order to focus on the essential, we use a single constant matrix $\bar{M}$, i.e. we do not need the topology-state s_t because the topology stays fixed. The $\bar{m}^{[j]}$ (which are the diagonal elements of $\bar{M}$) are chosen in such a way that it is beneficial to copy (share) the data and then transmit together, instead of broadcasting the data directly. Specifically, we set $\bar{m}^{[1]} = \frac{1}{4}$ and $\bar{m}^{[2]} = \bar{m}^{[3]} = 1$ and assume all links to be disjunct. Note that we can neglect $q^{[3]}$ in all further discussions since it merely represents the final destination.

With $\bar{M}$ being fixed, the efflux of the system can be expressed succinctly via the *control-option* $u_t = R\bar{M}v_t$ instead of the *control-vector* v_t. We define $\mathcal{U}$ to be the set of all possible control-options which is readily verified to be

$$\mathcal{U} = \{ \mathfrak{u}^0, \mathfrak{u}^1, \mathfrak{u}^2, \mathfrak{u}^3 \} = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 4 \end{pmatrix}, \begin{pmatrix} -4 \\ -4 \end{pmatrix} \right\}$$

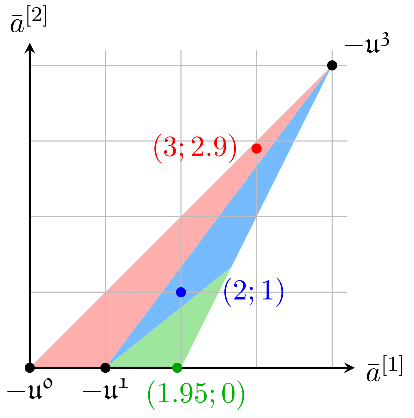
where we scaled all elements of $\mathcal{U}$ with the factor 4 to simplify the presentation. We have $\mathfrak{u}^1, \mathfrak{u}^2, \mathfrak{u}^3$ representing single transmission, data sharing, and joint transmission, respectively. And in each time-step the controller may only choose one of these options to influence the expected queue-state via $\mathbb{E}[q_{t+1} \,|\, q_t] = q_t + u_t + a_t$.

With this, it becomes very easy to express the set of all arrival rates $\bar{a} \in \mathcal{A}$ for which the system is possibly stabilizable:

$$\mathcal{A} = \left\{ \bar{a} \;\middle|\; \exists \bar{u} \in \operatorname{conv}(\mathcal{U}) : \bar{a} + \bar{u} < 0 \right\} \tag{2.43}$$

A graphical illustration can be found in Figure 2.7, where $\mathcal{A}$ corresponds to the green triangle on the left side.

Now, let us assume that there is no arrival at $q^{[2]}$, i.e. $\bar{a}^{[2]} = 0$. Using the control-options $\mathfrak{u}^2$ and $\mathfrak{u}^3$ in alternating sequence (given that there are enough packets to do so) would yield an efflux of 4 packets every 2 time-steps in $q^{[1]}$. Hence, an arrival rate just smaller than $\bar{a} = (2; 0)^\intercal$ can be stabilized and therefore belongs to $\mathcal{A}$. The corresponding point is referenced on the right side of Figure 2.7. It is easily checked that no other sequence of control-options can match this efflux.



Illustrated Arrival Rates
are stabilized by:

MW	(red)
fPNC$_{H=3}$	(blue + red)
PNC qPNC fPNC$_{H=2}$	(green + blue + red)

Fig. 2.8: Selection of Arrival Rates for the Simulation corresponding to Figure 2.9.

However, conventional back-pressure policies like MW are unable to access the control-option $\mathfrak{u}^2$, resulting in the loss of its throughput optimality in this example. It turns out that the only arrival rates stabilizable by MW are those in the red triangle on the right side of Figure 2.7.

In contrast, PNC is able to select the missing control-option $\mathfrak{u}^2$, and simulations suggest that it stabilizes the example for any strictly positive arrival rate $\bar{a}$ from $\mathcal{A}$. To substantiate this claim we refer to Figure 2.9. Here, we simulated the queue-state $q^{[1]}$ over time t for 3 different arrival rates $\bar{a}$ which are depicted in Figure 2.8 and initialized as scaled Bernoulli processes.

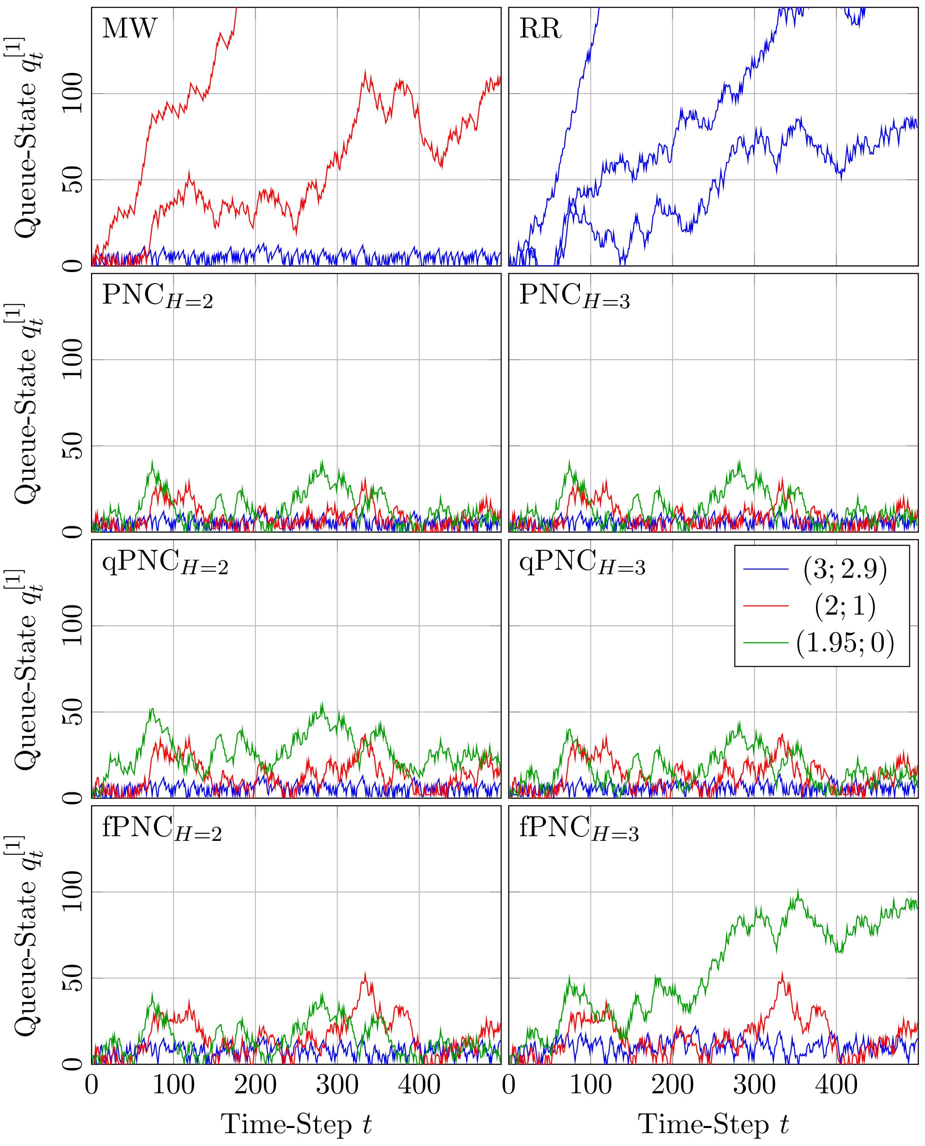


Fig. 2.9: Queue-state $q^{[1]}$ as a function of time t for various (color-coded) arrival rates, referring to the queueing network from Figure 2.6. MW, RR, and fPNC$_{H=3}$ seem to be inferior to the other predictive policies.

As predicted, MW does not stabilize the blue and green arrival rates (Figure 2.9). The RR policy naturally performs even worse since it is insensitive to queue-states and arrival rates, and therefore can only stabilize a very small set of arrival rates. As mentioned, fPNC can be interpreted as MW if time was dilated by the factor H. It performs better because it considers more than just the immediate next system-state but is only active every H time-steps. Surprisingly, fPNC performs worse when looking $H = 3$ steps ahead compared to $H = 2$. This can be attributed to the geometry of $\mathcal{A}$; a rigorous explanation is still subject to research. Finally, it can be seen that both PNC and qPNC stabilize all 3 arrival rates for all employed prediction horizons. This shows that (i) the MPC paradigm of repeating the optimization in each time-step is essential for optimal performance and (ii) the reduction of the quadratic objective to a linear one does not result in performance loss.

2.2.6 Computational Effort

Due to the consideration of multiple future system-states, PNC is nearly guaranteed to yield better control decisions compared to conventional back-pressure policies like MW. However, given that the optimization intrinsic to PNC is a binary linear program, we would expect that the effort to solve the optimization problem roughly grows exponentially with the horizon H since the feasible set of the optimization does so as well.

The simulation results seem to confirm this correlation. In Figure 2.10 we plotted the relative amount of time needed to find an optimal trajectory as a function of the horizon H. The results are normalized with respect to the value for $H = 1$ since this is the computation effort of the common back-pressure policy MW. Each value was generated by measuring the run-time over several thousand optimization runs from the presented examples in Section 2.2.5. Importantly, in our examples, queues are often times totally drained, resulting in many optimization parameters being 0 (especially for the objective vector). We call this the *unloaded* case. In order to escape this potential bias, we also performed the same measurements on simulations in which the initial queue-states were *loaded*. In these cases, we initialized the queue-states with a large amount of packets such that they could never be drained during the simulation. All simulations were performed using the well-established Gurobi Optimizer with Matlab.

For this investigation, we did not include the policies RR, fPNC and qPNC. The first one is not based on an optimization and therefore is not suited for comparison. The second one requires per definition the H-th part of the computational effort of PNC. Finally, we found that, in some cases, solving the optimization for qPNC takes more than 100 times as long as the optimization for PNC. Given that both policies yield nearly identical results, there seems to be no merit in investigating its computational effort any further.

In Figure 2.10, it can be seen that the additional effort to find an optimal trajectory, compared to MW, stays low for small values of H. E.g. for $H = 2$ the additional effort is just around 20% in the most unfavorable case of an unloaded initialization. For real world applications, we assume that this additional effort is even smaller since we expect networks to operate somewhere between the loaded and unloaded case.

Given the fact that significant performance gains can already be achieved by employing a horizon of $H = 2$, these results suggest that PNC is indeed a strong competitor to traditional back-pressure policies.

2.2.7 Conclusion

We equipped a discrete-time queueing network with an additional DTMC that changes network parameters (even topology) on a mid- to long-term time scale. We then introduced a novel family of predictive control policies, PNC, which is based on the paradigms of MPC, and devised a special implementation of the underlying prediction that allows the policy to be executed in the fastest way possible. Our policy is especially well suited to control the above mentioned systems and outperforms conventional control approaches, as is illustrated in numerical simulations, even if the prediction horizon is very small. Further simulations suggest that in those cases the additional computational effort required to run our policy is reasonably low. In

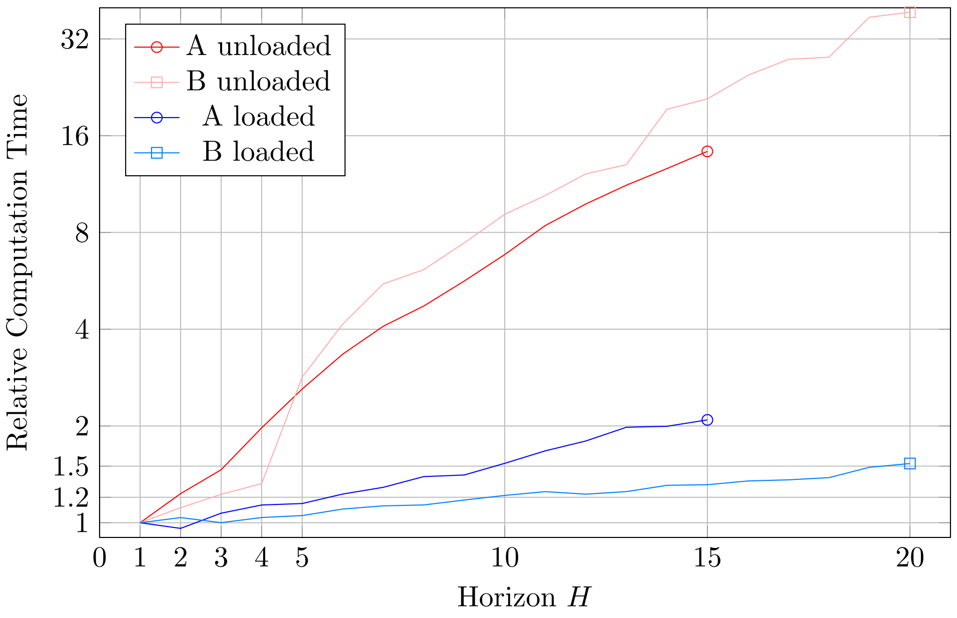


Fig. 2.10: Relative computation time as function of the (prediction-) horizon. A and B correspond to exemplary applications A and B from Section 2.2.5.

our main theoretical contribution, we prove throughput optimality of PNC. Looking ahead, we see an intriguing application in networks that consist of assembly-queues (e.g. found in parallel computing or manufacturing chains). Those networks still elude conventional control strategies but seem to be stabilizable under PNC policies with suitably chosen prediction horizon.

2.3 Concluding Remarks

The paper does not explain an interesting observation from Figure 2.9. Namely, that fPNC looses its ability to stabilize the arrival vector $(1.95; 0)$ as the horizon grows from $H = 2$ to $H = 3$. This contradicts the intuition that a larger prediction horizon yields better control performance.

The effect can be explained using an exemplary queueing network with the following control-options (this time explicitly considering the idle control-option as u^0):

$$\mathfrak{u}^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mathfrak{u}^1 = \begin{pmatrix} -4 \\ -4 \end{pmatrix}, \qquad \mathfrak{u}^2 = \begin{pmatrix} 0 \\ +2 \end{pmatrix} \tag{2.44}$$

The network is illustrated in Figure 2.11.

The stability region of the network is depicted in Figure 2.12, top left. It is readily verified (e.g. by means explained in Section 1.1) that MW can only ever use $\mathfrak{u}^1$ or $\mathfrak{u}^0$. MW chooses to activate $\mathfrak{u}^1$ when the system-state is in the red marked area in Figure 2.12, bottom left (of course, the red marked area extends to even larger system-states). For any other system-state, MW chooses to be idle ($\mathfrak{u}^0$) since negative system-states are ruled out. The repeated application of $\mathfrak{u}^0$ and $\mathfrak{u}^1$ only allows MW to stabilize a symmetric arrival with identical arrival sequences in each component.

Extending this investigation to fPNC with horizon $H = 2$ (which is equivalent to the MW policy considered over 2 time-steps because we follow the optimal control

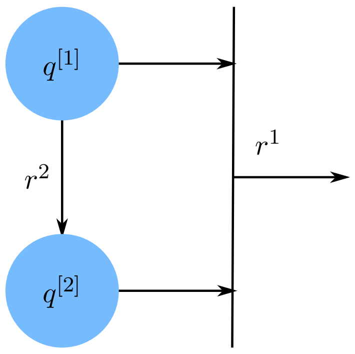


Fig. 2.11: Simple network example, corresponding to the control-options from (2.44). Link r^i results in control-option u^i.

trajectory until the end before starting a new optimization), we note that the amount of control-options is increased since any combination of two single-step control-options is now available to the policy, depicted in Figure 2.12, top mid. Crucially, that means that fPNC is able to activate $\mathfrak{u}^2$, albeit only in combination with $\mathfrak{u}^1$. This is possible when the system-state is located in the green marked area in Figure 2.12, bottom mid, and allows fPNC with horizon $H = 2$ to stabilize the entire positive stability region. (The question of whether this specific mapping from the queue-state to the control-options does indeed result in a throughput optimal behavior of the policy is exactly what the proof of throughput optimality is all about).

It becomes interesting once we extend the horizon once more and look at the fPNC policy with horizon $H = 3$. This time, all combinations of three single-step control-options are available to the policy. However, this makes the policy lose access to some part of the stability region (dark green area, Figure 2.12, top right), since it is no longer on the convex hull of all accessible control-options. Because though control-options like $(\mathfrak{u}^1 + 2\mathfrak{u}^2)$ are available in theory, fPNC will never use them (see the mapping in Figure 2.12, bottom right) and thus they do not contribute to the convex hull. This is the reason why fPNC loses parts of its stability region when increasing the horizon from $H = 2$ to $H = 3$.

Crucially, PNC does not lose its stabilizing abilities when increasing the horizon. Since the PNC policy does discard the entire tail of the optimized trajectory in every time-step, it is not bound to actually follow a control trajectory to its end. For PNC it is enough to gain access to $\mathfrak{u}^2$ via at least one combination of single-step control-options. If it does, it can indeed apply $\mathfrak{u}^2$ as often as is needed. Specifically, looking at Figure 2.12, bottom right, as long as the system-state stays in the green or orange area, PNC can activate $\mathfrak{u}^2$ because both $(\mathfrak{u}^1 + \mathfrak{u}^2 + 0)$ and $(2\mathfrak{u}^1 + \mathfrak{u}^2)$ give access to $\mathfrak{u}^2$. The order in which the control-options are activated in these trajectories is only influenced by the constraints.

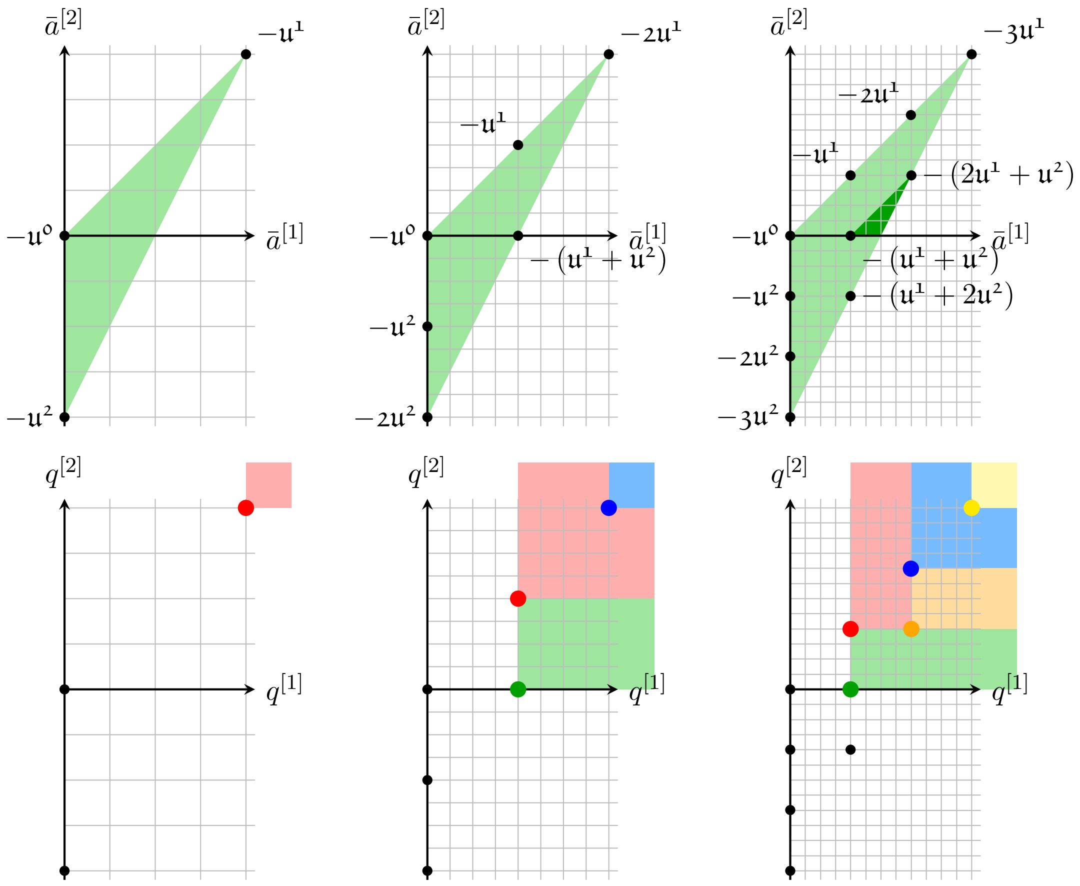


Fig. 2.12: Stability regions (top) and mapping from state-space to the control-options (bottom). The relocation of the (combined) control-options, caused by an increase of the horizon, is the reason why fPNC looses parts of its stability region when $H = 3$.

Paper 3

Assembly-Queues

Full Paper Title Detailed Stability Results for Assembly-Queues in Discrete-Time and Arbitrary Dimension

Publisher arXiv, 2020

Authors Richard Schöffauer, Gerhard Wunder

Acknowledgment This work is part of and thereby funded by the DFG Priority Program 1914.

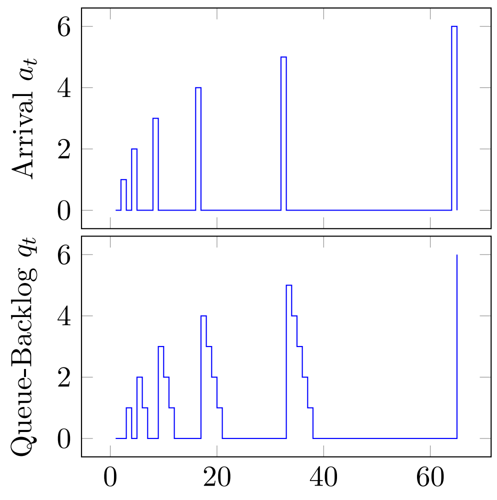
Abstract In a batch of assembly-queues, customers can only be served *all at once* or *not at all*, implying that service remains idle if at least one queue is vacant. As a common misconception, such batches are often deemed to be unstable, no matter the amount of queues employed. Our contribution consists of correcting such impreciseness: We prove that a batch of d assembly-queues in a discrete-time setting is quasi-stable (null-recurrent) for $d \in \{2, 3\}$ and unstable (transient) for $d \geq 4$. In order to present a concise proof, we assume all arrivals to our system to be Bernoulli processes, thereby avoiding the necessity of several technicalities. Nevertheless, our results can be extended to arbitrary proper stochastic processes in an obvious way. To facilitate our analytical results, we construct a correspondence between such queueing systems and a random-walk-like discrete-time Markov chain (DTMC) that operates on a quotient-space of the original state-space. For $d \geq 4$, transience can be shown by evaluating infinite power sums over skewed binomial coefficients. For the intricate case $d = 3$, we shall employ a theorem by Kendall which is a predecessor to the well-known recurrence theorem for DTMCs by Foster. Ignoring the special structure of the quotient-space, whose dimension is $d - 1$, our results correspond to Pólya's observation on random-walks.

3.1 Preliminary Remarks

The previous paper proved that the PNC policy is throughput optimal for conventional queueing networks. However, so is the MW policy (which is equal to PNC with Horizon $H = 1$). The real advantage of PNC compared to MW thus seems to be the fact that PNC retains the throughput optimality even in presence of so-called assembly queues. Those queues can only be served all at once or not at all, i.e. processing of packets/customers becomes impossible if at least one queue is empty. They were already employed in the examples related to Figure 1.2, Figure 1.6, and Figure 2.6 in the previous papers where they are easily identified. Given this observation, the paper at hand takes a closer look at the stability properties of such assembly queues.

In particular, the investigation revolves around the following somewhat counter-intuitive observation: Let there be a single queue in a time-discrete setting that can be drained by exactly one packet per time-slot (effectively decreasing its queue-backlog by 1) if the queue is not empty. At that queue, packets are arriving with arrival rate α (i.e. on average, the backlog increases by α packets per time-step). Under the usual assumption of an ergodic arrival process, it seems intuitive that the queue-backlog would be stable if $\alpha < 1$. And indeed this is the case. (To see why the arrival must be ergodic, we refer to Figure 3.1.)

Now let us extend this scenario by changing the queue into a batch of 2 assembly queues. As before, the system can be drained by 1 packet per time-step per queue, but only if both queues are non-empty. But now, each of the 2 queues is subject to a separate ergodic stochastic arrival process with arrival rates α_1 and α_2 (corresponding to the two queues). It is a reasonable assumption that if $\alpha_1 = \alpha_2 < 1$, i.e. if, on average, the same amount of packets are arriving at both queues, and if that average amount of packets is less than 1, then the queue-backlog is stable. However, this is not the case; one of the queue-backlogs almost surely grows to infinity. This behavior is investigated through a novel approach in the paper at hand.



From

$$a_t = \begin{cases} n & \text{for } t = 2^n \\ 0 & \text{else} \end{cases}$$

it follows that

$$\alpha = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} a_t = 0 < 1$$

but it also follows that

$$\lim_{n\to\infty} q_{2^n} \to \infty$$

Fig. 3.1: Necessity of ergodicity. The illustrated arrival process has an arrival-rate of 0, since the sum of all arrived packets grows sublinearly with time. Though the queue can be drained by 1 packet per time-step, the queue-backlog grows to infinity. This can only happen if the arrival process is not ergodic.

3.2 Paper Body

3.2.1 Introduction

Conventional queueing networks, consisting of queue/server *pairs*, are a well investigated system class. In those networks, each queue possesses its individual server which handles the customers exclusively for that queue. In contrast, there exists the notion of *assembly*-queues in which a single server is responsible for multiple queues. Here, service cannot take place if at least one queue is vacant, and successful service consumes one customer from each queue (see Figure 3.2). Assembly-queues find application in manufacturing processes [36], parallel computing [37], matchmaking scenarios [38] and communication [39].

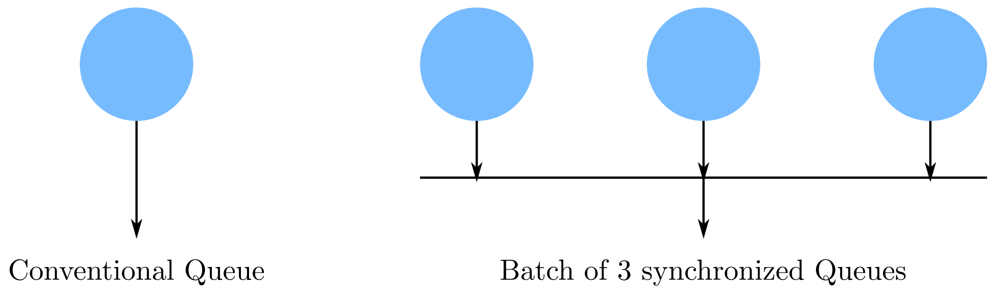


Fig. 3.2: Graphical illustration of a batch of 3 assembly-queues on the RHS; only if each queue has at least one packet queued up, service can be performed.

It is known that a single *batch* of assembly-queues with i.i.d. arrivals in a *continuous-time* setting is unstable in the sense that the backlog does not converge to a stationary distribution [40]. For the special case of 2 assembly-queues, this was further investigated by [41] who made a connection to the recurrence property of a corresponding Markov process and found that the difference between the queues, the so-called excess, is the decisive quantity which entails instability.

On the other hand, this intrinsic instability can be circumvented by enforcing finite buffer sizes or employing buffer-size-sensitive arrival processes. Using these techniques, subsequent research on assembly-queues roughly falls into 3 categories. i) The first one investigates properties like throughput or sojourn times of *individual* batches of assembly-queues [28, 29, 30]. ii) The second one is concerned with *autonomous networks* of assembly-queues. In such networks every server always operates as soon as its queue-states allow it to, hence a network controller is not required [31, 32, 33]. iii) The last category looks into *"shallow" networks* of assembly-queues. Here, many batches are arranged in parallel such that some queues are drained by multiple servers. A controller has to decide which batch to serve first, possibly preventing future service of other batches due to vacant queues. However, the network remains "shallow" in the sense that the efflux of one queue is never the influx of another [34, 35].

A common misconception is that batches of assembly-queues are unstable, regardless of how many queues are part of the batch. We correct this by proving that the behavior of the backlog significantly changes from null-recurrent to transient when employing more than 3 queues. Obviously, this can have decisive implications for those queueing networks in which batches of assembly-queues make up the core constituents.

In some sense, we extend the results from [41] to the multi-dimensional case and to a *discrete-time* setting. Compared to the findings of [40], we obtain a more precise characterization of the process that is responsible for the divergence of the backlog. We show that this process is the multi-dimensional excess which evolves on a specific quotient-space with equivalence classes taken from the general state-space. This process resembles a random-walk and will imply null-recurrence (quasi-stability) or transience (instability), based on the number of queues in the batch. However, significant differences (like the absence of "backward motion" in any dimension) prohibit us from directly employing the results from the theory of random-walks. Our claim is especially hard to prove for the case in which the batch consists of 3 assembly-queues. We will employ a theorem by Kendall [42] which can be seen as a predecessor to the famous recurrence theorem by Foster [43]. The other cases can be dealt with by evaluating infinite power sums over normalized binomial coefficients.

3.2.2 System Model

In the context of discrete-time queueing models, we can express a single batch of assembly-queues via the evolution of its queue-state. If the batch consists of d queues (*d*imensions), then the state-vector will be $q_t \in \mathbb{N}^d$, where t designates the time-step. Its process $\{q_t\}$ follows the evolution:

$$q_{t+1} = q_t - \mathbf{1}m_t v_t + a_t \tag{3.1}$$

The last term represents the multi-dimensional *a*rrival process $a_t \in \mathbb{N}^d$. We assume a_t to be a vector of d independent but identically parameterized Bernoulli processes with first moment α. I.e. in each time-step and each queue, the probability of exactly one customer arriving is α. While a_t represents the influx, $-\mathbf{1}m_t v_t$ represents the efflux, where *one* is the vector of ones (with dimension d) and $v_t \in \{0, 1\}$ is the control-vector. If $q_t \geq \mathbf{1}$, meaning that there is at least one customer present in each queue in time-step t, a controller can set $v_t = 1$ in order to serve the customers and thereby subtracting $\mathbf{1}$ from q_t. The sequence $\{m_t\}$ is an independent Bernoulli process with first moment μ that disturbs the control. I.e. only if $m_t = 1$, the activation of the control $v_t = 1$ has an effect on the system.

As mentioned, the control cannot be activated if at least one queue is vacant, resulting in the constraint

$$\mathbf{1}v_t \leq q_t \tag{3.2}$$

Obviously, an optimal control strategy activates v_t whenever possible and therefore only needs to depend on the current queue-state q_t (given that m_t is unknown prior to realization and the goal is to serve as many customers as possible). The maximum average efflux resulting from such a policy would be $\mathbf{1}\mu$. Furthermore, if $\alpha > \mu$, the influx a_t would on average be greater than the efflux $-\mathbf{1}m_t v_t$, rendering the system unstable no matter the control strategy. Hence, we will assume from now on that $\alpha < \mu$ (influx is smaller than maximum possible efflux) and an optimal control strategy is in place. The main contribution of this paper is the following theorem:

Theorem 2. *A batch of d assembly-queues, set up as described above, is quasi-stable (null-recurrent) for $d \in \{2, 3\}$ and unstable (transient) for $d \geq 4$.*

Proof. See Section 3.2.3 and Section 3.2.4. □

3.2.3 A Random-Walk on the Quotient-Space

Let the state-space of our queueing system be $\mathbb{Z}^d$ and identify the main diagonal of $\mathbb{Z}^d$ with the subspace $\langle \mathbf{1} \rangle := \{\, x = \mathbf{1} \cdot k,\ k \in \mathbb{Z} \,\} \subset \mathbb{Z}^d$.

The control decision v_t (as part of an optimal control strategy) only depends on the current queue-state q_t. It follows that with such a policy employed, the process $\{q_t\}$ becomes a discrete-time Markov chain (DTMC). Furthermore, $\{q_t\}$ can be divided into two distinct processes: a process $\{q_t^{\perp}\}$ perpendicular to $\langle \mathbf{1} \rangle$, and a process $\{q_t^{\shortparallel}\}$ parallel to $\langle \mathbf{1} \rangle$.

$$\begin{aligned} q_{t+1} &= \mathbf{1} q_{t+1}^{\shortparallel} + q_{t+1}^{\perp} \\ q_{t+1}^{\shortparallel} := q_{t+1} \operatorname{div} \mathbf{1} &= (q_t^{\shortparallel} + a_t) \operatorname{div} \mathbf{1} - m_t v_t \\ q_{t+1}^{\perp} := q_{t+1} \bmod \mathbf{1} &= \big(q_t^{\perp} + a_t\big) \bmod \mathbf{1} \end{aligned} \tag{3.3}$$

Here, $x^{\shortparallel} := x \operatorname{div} \mathbf{1}$ represents the maximum number of times that one can subtract $\mathbf{1}$ from x without a single entry of the resulting vector becoming negative. The remainder to x, which is $x - \mathbf{1}x^{\shortparallel}$, is $x^{\perp} := x \bmod \mathbf{1}$.

The process $\{q_t^{\shortparallel}\}$ represents the smallest queue in our initial queueing network and therefore operates on the 1-dimensional state-space $\mathbb{N}$. Note that $\{q_t^{\shortparallel}\}$ is *not* a DTMC (as shown in the proof of Proposition 1).

On the other hand, $\{q_t^{\perp}\}$ is readily verified to be a DTMC and can be identified as the multi-dimensional excess of customers, relative to the smallest queue. Crucially, $\{q_t^{\perp}\}$ is an autonomous process that cannot be influenced by the control. Due to the modulo operator, the state-space of $\{q_t^{\perp}\}$ is the quotient-space $\mathbb{Z}^d / \langle \mathbf{1} \rangle$. Though $\dim \big(\mathbb{Z}^d / \langle \mathbf{1} \rangle\big) = d-1$, we can still denote $\{q_t^{\perp}\}$ as a vector in $\mathbb{Z}^d$, keeping in mind that different vectors may represent the same equivalence class. This way, $\{q_t^{\perp}\}$ behaves similar to a random-walk: By the assumptions on a_t, the average probability of an increment per time-step in each entry of the vector q_t is α. However, regarding (3.3), there is no direct process to decrease any entry. Instead, by virtue of the quotient-space, a *decrement* in any entry is equal to an *increment* in *all other* entries:

$$\begin{pmatrix} -1 & 0 & \dots & 0 \end{pmatrix}^{\intercal} \equiv \begin{pmatrix} 0 & 1 & \dots & 1 \end{pmatrix}^{\intercal} \quad \bmod \mathbf{1} \tag{3.4}$$

We will investigate the recurrence property of $\{q_t^{\perp}\}$ in detail in Section 3.2.4.

Note that $\{q_t^{\shortparallel}\}$ and $\{q_t^{\perp}\}$ operate on *disjunct* state-spaces. Using a linear transformation on the canonical state-vector $q_t = \big(q_t^{[1]}, q_t^{[2]}, q_t^{[3]}\big)^{\intercal}$ yields a new description q_t' for the state-space in which the separation becomes obvious:

$$\begin{aligned} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} q_t &= \begin{pmatrix} q_t^{[1]} - q_t^{[3]} \\ q_t^{[2]} - q_t^{[3]} \\ q_t^{[3]} \end{pmatrix} \\ \begin{pmatrix} q_t^{[1]} - q_t^{[3]} \\ q_t^{[2]} - q_t^{[3]} \\ q_t^{[3]} \end{pmatrix} \operatorname{div} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} &= q_t^{[3]} \\ \begin{pmatrix} q_t^{[1]} - q_t^{[3]} \\ q_t^{[2]} - q_t^{[3]} \\ q_t^{[3]} \end{pmatrix} \bmod \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} &= \begin{pmatrix} q_t^{[1]} - q_t^{[3]} \\ q_t^{[2]} - q_t^{[3]} \\ 0 \end{pmatrix} \end{aligned} \tag{3.5}$$

This exact transformation is only valid as long as $q_t^{[3]}$ is the smallest entry in the canonical vector. As $\{q_t\}$ evolves, the transformation might have to be adapted accordingly. However, the separation will still hold true.

This clean separation of the state-spaces allows for the following claim:

Proposition 1. *Let $\{q_t\}$, $\{q_t^\perp\}$ and $\{q_t^{\shortparallel}\}$ be processes defined as above in* (3.1) *and* (3.3). *Then the following statements are true:*

1. *The process $\{q_t^{\shortparallel}\}$ is positive-recurrent.*
2. *If $\{q_t^\perp\}$ is positive-recurrent, null-recurrent or transient, then so is $\{q_t\}$.*

Proof. First, consider $\{q_t^{\shortparallel}\}$: If $q_t = \mathbf{0}$ ($\mathbf{0}$ being the zero-vector of dimension d), then $q_t^{\shortparallel} = 0$ and the probability of the event $\{q_{t+1}^{\shortparallel} = 1\}$ is α^d. If however $q_t = \mathbf{1} - \mathbf{e}_1$ ($\mathbf{e}_1$ being the first canonical unit vector of dimension d), then again $q_t^{\shortparallel} = 0$ but the probability of $\{q_{t+1}^{\shortparallel} = 1\}$ is α. Hence, $\{q_t^{\shortparallel}\}$ cannot be a DTMC. Let us introduce a dummy DTMC $\{w_t\}$ that also operates on $\mathbb{N}$ and whose transition probabilities, given that $w_t = \mathfrak{w}$, are

$$\left.\begin{aligned} \mathbb{P}[w_{t+1} = \mathfrak{w} + 1] &= \alpha \\ \mathbb{P}[w_{t+1} = \mathfrak{w} + 0] &= 1 - \alpha - \mu \\ \mathbb{P}[w_{t+1} = \mathfrak{w} - 1] &= \mu \end{aligned}\right\} \text{ if } \mathfrak{w} \geq 1$$
$$\left.\begin{aligned} \mathbb{P}[w_{t+1} = \mathfrak{w} + 1] &= \alpha \\ \mathbb{P}[w_{t+1} = \mathfrak{w} + 0] &= 1 - \alpha \end{aligned}\right\} \text{ if } \mathfrak{w} = 0 \tag{3.6}$$

and 0 otherwise. Hence, $\{w_t\}$ is an asymmetric random-walk and can be shown to be positive-recurrent via Foster's theorem [43], using an arbitrary linearly growing test-function. However, it is clear that, in any state, $\{w_t\}$ exhibits a higher probability to move further away from 0 than $\{q_t^{\shortparallel}\}$. Therefore, $\{q_t^{\shortparallel}\}$ must be positive-recurrent as well.

Now to statement 2: Since $\{q_t^{\shortparallel}\}$ is positive-recurrent, the process infinitely often visits any coherent subset of $\mathbb{N}$ that contains 0, and the time until it revisits such a set is finite almost surely. Conversely, when picking infinitely many values from a realization of $\{q_t^{\shortparallel}\}$, the probability of these values lying inside a finite set is 1 a.s. .

We know that $\{q_t^{\shortparallel}\}$ takes the value of the smallest queue, and that $\{q_t^\perp\}$ represents the differences between this smallest queue and all other queues. If $\{q_t^\perp\}$ is transient, the differences and therefore the queues themselves actively grow to infinity, making $\{q_t\}$ transient as well. If $\{q_t^\perp\}$ is non-transient, the event {all differences return to 0 at the same time} happens infinitely often. If we look at all the values of $\{q_t^{\shortparallel}\}$ in these events, we already argued that these values belong to some finite set $\mathcal{S} \subset \mathbb{N}$ a.s. . Hence, $\{q_t\}$ also revisits $\mathcal{S}$ (embedded in $\mathbb{Z}^d$) infinitely often, and each time it does, all queues have the same backlog. It follows that $\{q_t\}$ must be non-transient. The expected time it takes to revisit $\mathcal{S}$ clearly is either infinite or finite, depending on whether $\{q_t^\perp\}$ is null- or positive-recurrent. The proposition follows. □

Notably, since $\{q_t^\perp\}$ is autonomous, it follows from Proposition 1 that the stability property of a batch of assembly-queues is fully independent of the control (remember that we assumed $\alpha < \mu$). With this we can reformulate Theorem 2 as

Theorem 3. *The process $\{q_t^\perp\}$ is null-recurrent for $d \in \{2,3\}$ and transient for $d \geq 4$.*

Proof. See Section 3.2.4 □

3.2.4 Proof of Main Theorem

We separate the proof into 3 parts for the cases $d = 2$, $d = 3$ and $d \geq 4$. The following lemma, holding 2 separate results, will be useful.

Lemma 4. *Let there be a set of elements $\mathcal{X} := \{x_1, x_2, \dots x_n\}$, $x_i \in \mathbb{R}_+$ with $\sum_{i=1}^n x_i = 1$. Denote the maximal value with $\hat{x} = \max_{i=1,\dots n} x_i$ and the average with $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$. Then it holds for each $d \in \mathbb{N}, d \geq 4$ that*

$$\sum_{i=1}^n x_i^d \leq \hat{x}^{d-1} \qquad \text{and} \qquad \sum_{i=1}^n x_i^2 \geq n\bar{x}^2 \tag{3.7}$$

Proof. It is

$$\sum_{i=1}^n x_i^d \leq \sum_{i=1}^n x_i \hat{x}^{d-1} = \hat{x}^{d-1} \tag{3.8}$$

and

$$\sum_{i=1}^n x_i^2 = \sum_{i=1}^n (\bar{x} + [x_i - \bar{x}])^2 = n\bar{x}^2 + \underbrace{2\bar{x}\sum_{i=1}^n [x_i - \bar{x}]}_{=0} + \sum_{i=1}^n [x_i - \bar{x}]^2 \geq n\bar{x}^2 \tag{3.9}$$

□

Case $d \geq 4$

Per assumption all arrival processes are Bernoulli processes with parameter α. We denote the probability of such a process exhibiting k increments in n time-steps as $p_{n,k}$ and have

$$p_{n,k} = \binom{n}{k} \alpha^k (1-\alpha)^{n-k} \tag{3.10}$$

Let r_n be the probability that $\{q_t^\perp\}$ has *r*eturned to wherever it originated from after n steps. It is well known that $\{q_t^\perp\}$ is transient if the infinite sum $\sum_{n=1}^\infty r_n$ converges, and non-transient if it diverges. Because of the modulo operator, $\{q_t^\perp\}$ has returned in n steps if all queues have experienced the same amount of increments. Hence, r_n becomes

$$r_n = \sum_{k=0}^n p_{n,k}^d = \sum_{k=0}^n \left[\binom{n}{k} \alpha^k (1-\alpha)^{n-k}\right]^d \tag{3.11}$$

We can identify $p_{n,k}^d$ with a position in Pascal's triangle: n denotes the row and k the column. (Especially for $\alpha = \frac{1}{2}$ the sum $\sum_{n=1}^\infty r_n$ runs over the elements of the normalized triangle, taken to the power of d.) For ease of notation we will use the identity $\beta = 1 - \alpha$ and will treat $p_{n,k}^d$ as a function of α when necessary.

It is

$$\sum_{k=0}^{n} p_{n,k} = 1 \qquad \forall n \in \mathbb{N},\ \forall \alpha \in (0,1) \tag{3.12}$$

The terms in row n, which are $p_{n,0}, \dots p_{n,n}$, follow a binomial distribution which exhibits 2 maxima at most. Denote with $\hat{p}_n$ the largest term in row n, and its position with $\hat{k}_n$. Going through the terms of the row n from $k = 0$ to $k = n$ (left to right), $\hat{p}_n$ is positioned wherever the quotient of two consecutive terms is greater or equal 1 for the first time:

$$\frac{\binom{n}{\hat{k}_n}\alpha^{\hat{k}_n}\beta^{n-\hat{k}_n}}{\binom{n}{\hat{k}_n+1}\alpha^{\hat{k}_n+1}\beta^{n-\hat{k}_n-1}} \geq 1 \quad \Longleftrightarrow \quad \hat{k}_n \geq n\alpha - \beta \tag{3.13}$$

Due to the discreteness, we can only locate $\hat{k}_n$ down to an interval such that $\hat{k}_n = n\alpha + \delta$ with $-\beta \leq \delta \leq \alpha$.

Using Stirling's approximation $\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n} \leq n! \leq en^{n+\frac{1}{2}}e^{-n}$ and some basic algebra, we obtain the value of the largest term in a row as

$$\begin{aligned}\hat{p}_n &= \binom{n}{n\alpha+\delta}\alpha^{n\alpha+\delta}\beta^{n\beta-\delta} \\ &\leq \frac{e}{2\pi}\frac{1}{\sqrt{n\alpha\beta}}\left(1+\frac{\delta}{n\alpha}\right)^{-n\alpha-\delta-\frac{1}{2}}\left(1-\frac{\delta}{n\beta}\right)^{-n\beta+\delta-\frac{1}{2}}\end{aligned} \tag{3.14}$$

Regarding the last 2 factors, we can make the following estimation and find, for any given $\varepsilon \in \mathbb{R}_+$, an $N \in \mathbb{N}$ such that $\forall n \geq N$:

$$\begin{aligned}\left(1+\frac{\delta}{n\alpha}\right)^{-n\alpha} &\to e^{-\delta} && \leq e^{\beta} + \varepsilon \\ \left(1+\frac{\delta}{n\alpha}\right)^{-\delta} & && \leq 1 \\ \left(1+\frac{\delta}{n\alpha}\right)^{-\frac{1}{2}} &\to 1 && \leq 1 + \varepsilon\end{aligned} \tag{3.15}$$

With slight abuse of notation concerning the values of ε and N, this yields

$$\hat{p}_n \leq \frac{e^2+\varepsilon}{2\pi}\frac{1}{\sqrt{n\alpha\beta}} \qquad \forall n \geq N \tag{3.16}$$

Now, for any d let s_d denote the finite sum of the first $N-1$ rows, i.e.

$$s_d = \sum_{n=1}^{N-1} r_n = \sum_{n=1}^{N-1}\sum_{k=0}^{n} p_{n,k}^d \tag{3.17}$$

Using Lemma 4, the entire series becomes

$$\begin{aligned}\sum_{n=1}^{\infty}\sum_{k=0}^{n} p_{n,k}^d = s_d + \sum_{n=N}^{\infty}\sum_{k=0}^{n} p_{n,k}^d &\leq s_d + \sum_{n=N}^{\infty}\hat{p}_n^{d-1} \\ &\leq s_d + \left[\frac{e^2+\varepsilon}{2\pi\sqrt{\alpha\beta}}\right]^{d-1}\sum_{n=N}^{\infty}\frac{1}{n^{\frac{d-1}{2}}}\end{aligned} \tag{3.18}$$

which converges for $d \geq 4$. This proves transience of $\{q_t^{\perp}\}$ for $d \geq 4$.

Case $d = 2$

Again, identify $p_{n,k}$ with its corresponding position in Pascal's triangle (row n, column k). Every row consists of $n+1$ terms which add up to $(\alpha+\beta)^n = 1$ (using the binomial theorem). If we average the values of each row, we get an average value of $\frac{1}{n+1}$. Hence, using Lemma 4 yields

$$\sum_{n=1}^{\infty} r_n = \sum_{n=1}^{\infty}\sum_{k=0}^{n} p_{n,k}^2 \geq \sum_{n=1}^{\infty}\sum_{k=0}^{n} \frac{1}{(n+1)^2} > \sum_{n=1}^{\infty} \frac{1}{(n+1)} \tag{3.19}$$

which diverges. It follows that $\{q_t^{\perp}\}$ is non-transient for $d = 2$. To answer the question, whether in this case $\{q_t^{\perp}\}$ is positive-recurrent or null-recurrent, we investigate the *E*xpected *T*ime of the process's *F*irst *R*eturn to wherever it started (ETFR).

For that purpose, define the process

$$h_t = q_t^{[1]} - q_t^{[2]} = q_0^{[1]} - q_0^{[2]} + \sum_{\tau=0}^{t-1} a_\tau^{[1]} - a_\tau^{[2]} \tag{3.20}$$

It is readily verified that $\{h_t\}$ is a simple symmetric random-walk in 1 dimension with the additional possibility of a "null"-increment occurring. Hence, the ETFR of $\{h_t\}$ must be greater than that of a simple symmetric random-walk, and therefore the ETFR of $\{h_t\}$ is infinite.

However, we can write $\{q_t^{\perp}\}$ as

$$q_t^{\perp} = h_t \cdot \begin{cases} \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \text{if } q_t^{[1]} > q_t^{[2]} \\ \begin{pmatrix} 0 \\ -1 \end{pmatrix} & \text{if } q_t^{[1]} \leq q_t^{[2]} \end{cases} \tag{3.21}$$

and hence, the ETFR of $\{q_t^{\perp}\}$ must be the one of $\{h_t\}$. This proves null-recurrence of $\{q_t^{\perp}\}$ for $d = 2$.

Case $d = 3$

This case is more intricate. In a first step we obtain a lower bound on the series $\sum_{n=1}^{\infty} r_n$ by setting $\alpha = \beta = \frac{1}{2}$. It is readily verified that

$$\frac{d}{d\alpha} r_n(\alpha) = 0 \quad \text{and} \quad \frac{d^2}{d\alpha^2} r_n(\alpha) > 0 \qquad \text{if} \quad \alpha = \frac{1}{2} \tag{3.22}$$

and no other candidates for extreme points exist. Hence, it will suffice to show divergence for $\alpha = \frac{1}{2}$, and we will make use of this substitution in subsequent formulas.

Note that it is also true that the series $\sum_{n=1}^{\infty} r_n$ diverges if the DTMC $\{q_t^{\perp}\}$ is non-transient (we only used the converse before). And for $\alpha = \frac{1}{2}$, we can show the latter part by applying a theorem by Kendall [42] (a predecessor to the famous recurrence theorem by Foster [43]): The DTMC $\{q_t^{\perp}\}$ with state-space $\mathcal{Q}$ is non-transient, iff there exists a function $f : \mathcal{Q} \to \mathbb{R}_+$ and a finite set $\mathcal{F} \subset \mathcal{Q}$ such that

$$\begin{aligned} \mathcal{Q}_K &:= \{\, \mathfrak{q} \in \mathcal{Q} : f(\mathfrak{q}) \leq K \,\} = \text{finite for all } K \in \mathbb{R}_+ \\ \Delta f(\mathfrak{q}) &:= \mathbb{E}[q_{t+1} - q_t \,|\, q_t = \mathfrak{q}] \leq \infty \qquad \forall \mathfrak{q} \in \mathcal{Q} \\ \Delta f(\mathfrak{q}) &:= \mathbb{E}[q_{t+1} - q_t \,|\, q_t = \mathfrak{q}] < 0 \qquad \forall \mathfrak{q} \in \mathcal{Q} \setminus \mathcal{F} \end{aligned} \tag{3.23}$$

Here, the gothic q, which is $\mathfrak{q}$, is used to denote possible realization of the random variables that the process $\{q_t\}$ is made of.

Next, we show that this theorem is fulfilled for the function

$$f(\mathfrak{q}) = \ln\ln(e + \rho(\mathfrak{q})) \tag{3.24}$$

where e is Euler's number and $\rho(\mathfrak{q})$ is the squared distance from $\mathfrak{q}$ to the main diagonal of the original state-space $\mathbb{Z}^3$:

$$\rho(\mathfrak{q}) = \mathfrak{q}^\intercal \mathfrak{q} - \frac{1}{3}\left(\mathfrak{q}^\intercal \mathbf{1}\right)^2 \tag{3.25}$$

It is also readily checked that this function is well defined for the actual state-space $\mathcal{Q} = \mathbb{Z}^3/\langle \mathbf{1} \rangle$ of $\{q_t^\perp\}$ and fulfills the first condition in (3.23). The drift $\Delta f(\mathfrak{q})$ is the expected change in f during one evolution of $\{q_t^\perp\}$ and is independent of the control, a feature inherited from $\{q_t^\perp\}$. Because the entries in a_t are from the set $\{0,1\}$, and by the virtue of $\alpha = \frac{1}{2}$, the drift simply becomes

$$\begin{aligned}\Delta f(\mathfrak{q}) = -f(\mathfrak{q}) &+ \frac{1}{8}\left[f(\mathfrak{q}+\mathbf{e}_1+\mathbf{e}_2+\mathbf{e}_3) + f(\mathfrak{q})\right]\\ &+ \frac{1}{8}\left[f(\mathfrak{q}+\mathbf{e}_1) + f(\mathfrak{q}+\mathbf{e}_2) + f(\mathfrak{q}+\mathbf{e}_3)\right]\\ &+ \frac{1}{8}\left[f(\mathfrak{q}+\mathbf{e}_1+\mathbf{e}_2) + f(\mathfrak{q}+\mathbf{e}_2+\mathbf{e}_3) + f(\mathfrak{q}+\mathbf{e}_3+\mathbf{e}_1)\right]\end{aligned} \tag{3.26}$$

with $\mathbf{e}_i$ being the i-th canonical basis vector.

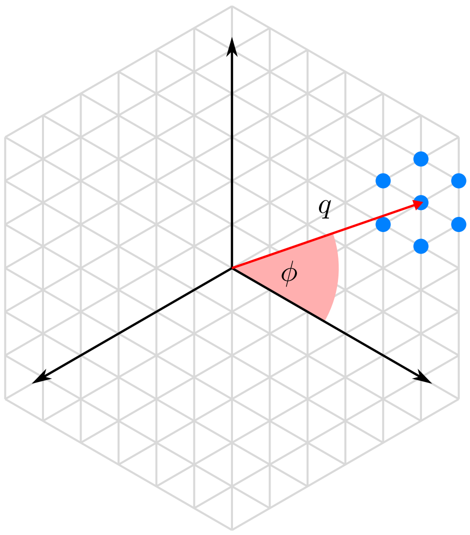


Fig. 3.3: State-space $\mathbb{Z}^3/\langle \mathbf{1} \rangle$; r and ϕ describe a current queue-state $q_t^\perp$, while the points mark all possible states that $q_{t+1}^\perp$ can inhabit.

In our case, the fastest way to verify Kendall's theorem is via a geometrical interpretation of the state-space. For $d = 3$, we essentially move on a 2-dimensional plane with 3 axis as illustrated in Figure 3.3. Hence, we can identify $\mathfrak{q}$ with its radius r and an angle ϕ shared with the projection of the $(1,0,0)$ axis. A third coordinate that would hold information about the distance between $\mathfrak{q}$ and the origin parallel to the main diagonal is not required because it would get "absorbed" by $\rho(\mathfrak{q})$. In these coordinates, and using the cosine formula, the drift becomes

$$\Delta f(\mathfrak{q}) = -\frac{6}{8}\ln\ln\left(e+r^2\right) + \frac{1}{8}\sum_{m=0}^{5}\ln\ln\left(e+r^2+1-2r\cos(\phi+m\cdot 60°)\right) \quad (3.27)$$

Differentiating by ϕ yields maxima at $\phi = 30° + z\cdot 60°$, with $z\in\mathbb{Z}$, such that

$$\begin{aligned}\Delta f(\mathfrak{q}) \leq -\frac{3}{4}\ln\ln\left(e+r^2\right) + \frac{1}{4}\ln\ln\left(e+r^2+1\right) \quad (3.28)\\ +\frac{1}{4}\ln\ln\left(e+r^2+1-\sqrt{3}r\right) + \frac{1}{4}\ln\ln\left(e+r^2+1+\sqrt{3}r\right)\end{aligned}$$

The RHS obviously tends to 0 as $r\to\infty$ because r^2 dominates the other terms in the ln-function, and the coefficients in front of ln-functions add up to 0. On the other hand, differentiating the RHS by r yields

$$-\frac{2}{r^2\ln\left(r^2\right)} + \frac{1}{\left(r^2-\sqrt{3}r\right)\ln\left(r^2-\sqrt{3}r\right)} + \frac{1}{\left(r^2+\sqrt{3}r\right)\ln\left(r^2+\sqrt{3}r\right)} > 0 \quad (3.29)$$

which is positive due to the convex nature of the occurring functions as $r\to\infty$. This allows us to deduct as follows: in the direction ϕ of the largest possible values for $\Delta f(\mathfrak{q})$, the value of $\Delta f(\mathfrak{q})$ tends to 0 as r grows to infinity. Furthermore, since the derivation by r is positive, the value tends to 0 but is always negative if r is large enough. It follows that $\Delta f(\mathfrak{q})$ must be negative for all $r\in\mathbb{R}_+$ beyond a certain threshold, leaving only a finite set of states for which $\Delta f(\mathfrak{q})\geq 0$, and thus fulfilling (3.23). Hence, $\{q_t^\perp\}$ is non-transient if $d=3$.

To investigate whether in this case $\{q_t^\perp\}$ is positive- or null-recurrent, we employ the same technique as in case $d=2$. Therefor we define the processes

$$h_t^{[ij]} = q_t^{[i]} - q_t^{[j]} = q_0^{[i]} - q_0^{[j]} + \sum_{\tau=0}^{t-1} a_\tau^{[i]} - a_\tau^{[j]} \qquad \text{for } i,j\in\{1,2,3\}, i\neq j \quad (3.30)$$

each of which is again a simple symmetric random-walk with additional "null"-increment, and exhibits an infinite ETFR (see case $d=2$). And because we can express $\{q_t^\perp\}$ through

$$q_t^\perp = \begin{cases}\begin{pmatrix}h_t^{[1,3]}\\ h_t^{[2,3]}\\ 0\end{pmatrix} & \text{if } q_t^{[3]} < q_t^{[1]}, q_t^{[2]}\\ \begin{pmatrix}h_t^{[1,2]}\\ 0\\ h_t^{(3,2)}\end{pmatrix} & \text{if } q_t^{[2]} < q_t^{[3]}, q_t^{[1]}\\ \begin{pmatrix}0\\ h_t^{[2,1]}\\ h_t^{[3,1]}\end{pmatrix} & \text{if } q_t^{[1]} \leq q_t^{[2]}, q_t^{[3]}\end{cases} \quad (3.31)$$

the ETFR of $\{q_t^\perp\}$ must also be infinite, making $\{q_t^\perp\}$ null-recurrent for the case $d=3$. □

3.2.5 Conclusion

We prove that a batch of d time-discrete assembly-queues with i.i.d. arrivals, operating under an optimal control, is quasi-stable for $d \in \{2, 3\}$ and unstable for $d \geq 4$. Here, quasi-stability and instability refer to null-recurrence and transience of the corresponding DTMC that represents the multi-dimensional excess of the queues, respectively. This implies that for $d \in \{2, 3\}$, some queue-states possibly become infinite in some time-steps, but will *not* actively grow towards it (in contrast to the queue-states in the transient cases $d \geq 4$). In light of the fact that said DTMC evolves in dimensions $d - 1$ and resembles a random-walk on a quotient-space, our result is in accordance with the well-known observation by Pólya [44], stating that a symmetric random-walk is null-recurrent in 1 or 2 dimensions and transient in 3 or more dimension.

3.2.6 Open Questions

As already investigated for the 2-dimensional case in continuous-time by [41], it stands to question, whether the queueing process becomes stable if the arrival process is assumed to be semi-stochastic. E.g. one could assume that in every consecutive interval of $T \gg 0$ time-steps, the same amount of customers has arrived at each queue. Another option would be to slightly in- or decrease the arrival rates based on the excess in a manner that minimizes it.

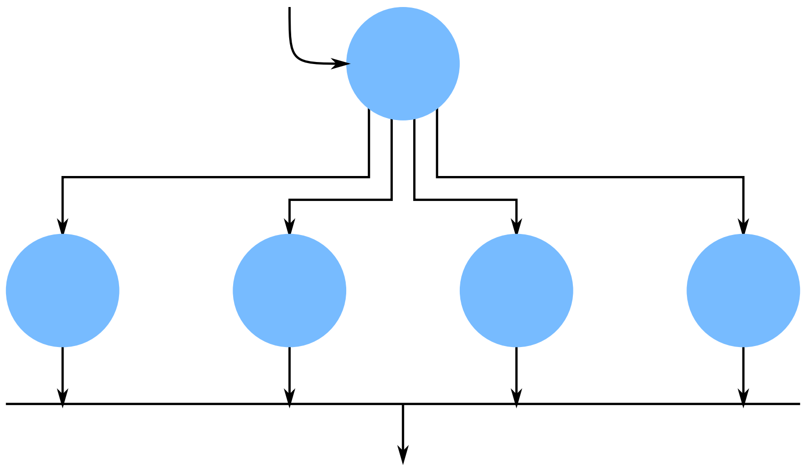


Fig. 3.4: Network, consisting of a conventional queue, that redirects its customers to different queues of a batch of assembly-queues.

Furthermore, in a network context, the stochastics from the arrival processes might be mitigated entirely before they reach the batch. See e.g. Figure 3.4 where a controller can decide which of the assembly-queues to feed, while the only stochastic arrivals happen to the upper queue.

Finally, (though we assume it to be true) it stands to show that our results can be generalized to the continuous-time model and to the case in which service of the batch requires more than one customer at certain queues (asymmetric batch-throughput).

3.3 Concluding Remarks

The overall contribution of this paper is somewhat limited by the fact that the final result, the instability of batches of queues, is already known. However, it was only investigated for a slightly different set-up in which queues have probabilistic processing times instead of having to be activated by an all-knowing controller. Nevertheless, to the best of our knowledge, we are the first to treat random walks on quotient spaces and show their applicability to this specific subject.

Paper 4

Quadratic Reliable Prediction

Full Paper Title Distributed MPC with Prediction of Time-Varying Communication Delay

Publisher IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys), 2018

Authors Jannik Hahn, Richard Schöffauer, Gerhard Wunder, Olaf Stursberg

Acknowledgment Partial financial support by the DFG within the Priority Program *Cyber-Physical Networking* (SPP 1914) is gratefully acknowledged.

Abstract The novel idea presented in this paper is to interweave distributed Model-Predictive Control with a reliable scheduling of the information that is interchanged between local controllers of the plant subsystems. To this end, a dynamic model of the communication network and a predictive scheduling algorithm are proposed, the latter providing predictions of the delay between sending and receiving information. These predictions can be used by the local subsystem controllers to improve their control performance, as exemplary shown for a platooning example.

4.1 Preliminary Remarks

As showcased in the previous papers, a predictive control policy features certain benefits compared to a conventional myopic policy. However, the overarching goal of the project was not merely to improve the control of queueing networks via predictions, but to investigate how such predictions in the communication realm might benefit the control realm (both realms being intertwined in a Cyber-Physical System). The paper at hand takes a first approach in this direction, using cooperative robust MPCs in the control realm. Robust MPCs are controllers that guarantee tight bounds on the control performance even under uncertain disturbances. To guarantee this, though, they require these disturbances to be tightly bounded themselves (because, naturally, no controller is able to correct for disturbances with infinite amplitude). Considering that communication delay in the control loop simply equals another disturbance and therefore requires the delay to be bounded, this poses a problem. It stands in opposition to our model of the communication system so far since we model transmission failure via Bernoulli trials and thus cannot guarantee a certain worst-case delay that is never exceeded. (There is always a finite probability that a given number of consecutive Bernoulli trials fails).

Short of changing the communication model, the only way to bridge the gap between the control side (that requires fixed worst-case delays) and the communication side (that cannot guarantee any worst-case delays) is to define a probability threshold and neglect any delay values that occur with probability smaller than that threshold. Though this seems like acting as if the problem doesn't exist, this is actually quite a normal approach in practice since one can never rule out failure with absolute certainty. Still, this method entails a lot of new problems for the network control policy that we try to address in the following paper.

Please note that the paper deviates from the previously used notation by using the variable r to express a certain amount of repetitions rather then letting it refer to a vector in the routing matrix (i.e. a link). Also, it is implicitly assumed that the Heavyside function $\Theta[x]$ is defined such that $\Theta[0] = 1$ (while usually, $\Theta[0] = 0.5$).

4.2 Paper Body

4.2.1 Introduction

Two major trends can be recognized in the modern information society: one is that more and more physical systems used on a daily basis are equipped with digital controllers, sensors, and actuators (leading to *embedded systems*) – secondly, these embedded systems are connected to a global information network (the *cyber space*). Bringing these two trends together is a current main challenge in engineering. Systems in which communication and control are formulated within a common mathematical model are called *Cyber-Physical-Systems* (CPS), see [45]. In CPS, the traditional modeling of plant and controller is extended by a model of the (wireless) communication between actuators, sensors, and multiple control units. In the standard setting, the overall objective remains in the realm of control, while stringent requirements are formulated for the communication. Of crucial importance is the latency in the wireless network, i.e. the delay of a packet that propagates through the communication system from the source to its destination. Since control usually implies a closed-loop setting, this delay gives a lower bound on how fast the controller may react to any system changes, thus limiting its effect on the system. Obviously this has a negative and even nonlinear effect on the control performance.

The most intuitive and typical constraint is that the worst case delay has to be smaller than the time interval with which the controller operates. If so, the controller does not experience any delay. A small worst case delay potentially allows for greater clock rates, leading to higher control performance. While there exists a good understanding of how such delay influences the control system [46] [47], only few ideas have emerged that go beyond such simple models. In [48], a co-design of communication and control is proposed that allows for a distributional relaxation of delay constraints, such that the worst case delay may be larger than the controllers' time interval of operation. Another currently investigated idea evolves around event-based control schemes, helping to lower the requirements on the communication system while maintaining control performance [49], [50]. Additionally, a protocol design, specifically tailored to the control needs, can lower effects of jitter and packet loss [51], leading also to delay minimization.

From the control perspective, achieving common control objectives for various autonomous and dynamically (de-) coupled subsystems is a challenge, in particular if input and coupled or uncoupled state constraints have to be satisfied. In [52], an approach of distributed Model-Predictive Control (DMPC) for this purpose was introduced, but communication of subsystems and delayed information exchange are neglected. In [53] and [54], DMPC schemes are proposed by addressing constant communication delays. Time-varying delay is treated in [55] according to a switching topology of feedback controllers, while neglecting input and state constraints. All these approaches aim at enabling a maximum control frequency for a given delay, or improving the control result under constant or worst case delay.

In this paper, we explore a different approach: The main idea is that the control algorithm takes *reliable* packet delay predictions from the communication system, and gives this information to the corresponding plant controllers. Hence the controller is provided with the additional information, of when an expected set of data packets will arrive. The main contribution of this paper is to show: 1) how to create these delay predictions in the communication system and 2) how to use them in the control system to improve performance. Interestingly, to predict packet delay over some finite horizon is, to the best of our knowledge, a novel approach not been considered before.

4.2.2 General Approach

The entire system-model consists of two major parts, see Fig. 4.1: First, in a distributed setting, multiple physical subsystems (plants) are each equipped with a local controller. These subsystems are assumed to be dynamically decoupled, but are coupled through a common control objective to perform a cooperative task. To achieve this objective, the plant controllers exchange information (like state and input trajectories) over a communication system subject to time-delay. Receiving a delayed state and input trajectory, an uncertain nominal state trajectory of the sending subsystem can be reconstructed by knowing the plant dynamics of the sender. Having an upper bound of possible uncertainty caused by communication delay and of possible deviations from previous communicated predictions, each subsystem can use robust MPC (RMPC) with respect to these uncertainties. By repeatedly solving local RMPC optimization problems for each subsystem, the common control goal can be achieved.

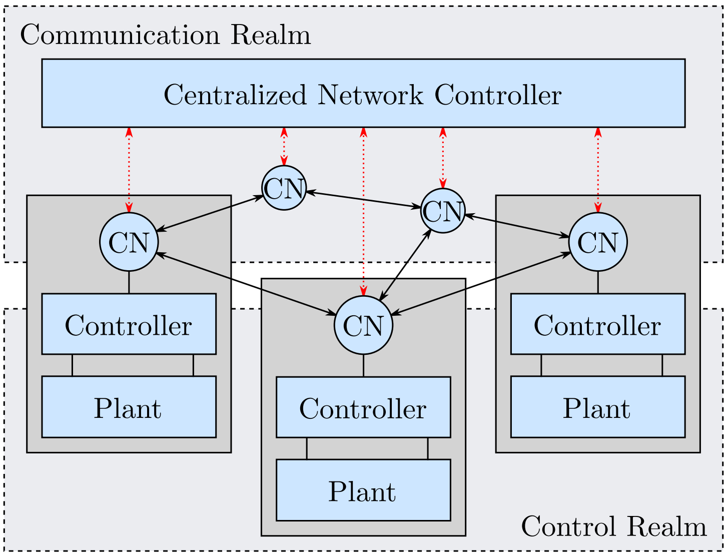


Fig. 4.1: General structure and communicated information. Each plant controller represents a node (CN) of the network. While communication between nodes and network controller is instantaneous, communication between nodes is modeled by a queueing network.

The underlying communication system itself consists of multiple communication nodes (CN), which are not necessarily all located at the local controllers. The data is routed over the communication system, and the routing is controlled by a dedicated, centralized network controller. The functionality of the overall model is as follows:

1. The network controller not only routes and schedules the data flows generated by the plant controllers over multiple hops in the network, but is designed to also predict packet delays over a future horizon.

2. This prediction contains information about when data will arrive at the plant controllers with (to a certain degree adjustable) reliability. Reliability is achieved through simple repetition of sending a packet in a hop. This leads to so-called *delay trajectories*, which are communicated in each time instance to the plant controllers (red arrows in Fig. 4.1).

3. The plant controllers are designed to exploit these *delay trajectories* locally by RMPC to achieve an improved common control performance.

To exchange data, each controller acts as a communication node (in a possibly larger network). The dedicated network controller has knowledge of all nodes with their states, of the momentary network topology (meaning accessible communication links between the nodes), and of where the data is scheduled to arrive. Furthermore, the network controller is implemented as an MPC, too. Only then, we are able to yield future communication schedules, which are necessary for predicting delay times of exchanged data by the plant controllers. Note that within this scheme, we assume that transmission along the communication network is time-consuming, however the network controller can send its predicted delay trajectories as well as its control inputs without any delay directly to the communication nodes and associated plant controllers. This is a valid assumption as long as the size of data, exchanged by the plant controllers, is considerably larger then the size of the delay trajectories.

The next two sections will go into the details of the communication and control system design.

4.2.3 Communication System

We model the communication network with a discrete-time, packet-based queueing system. Herein, each network entity i is assigned to one queue q^i, which is 1, if a corresponding packet is present at the entity, and 0 otherwise. Hence, packet transmission can be modeled as increasing (and decreasing) queues by 1. We denote all queues as a *queue vector* $q = (q^{[1]} \dots q^{[n_q]})^\intercal \in \{0,1\}^{n_q}$, where n_q is the number of entities in the system. If entity i wants its information to be transported to some destination (a *communication request*), the queuing system is initialized with q^i being 1 while all other entries in q are 0. The system evolves with t according to:

$$q_{t+1} = q_t + RM_t v_t. \tag{4.1}$$

Entities can exchange packets if there exists at least one communication *link* between them. A link is a vector, which adds 1 to a queue (and possible subtracts 1 from another). They are collected as columns in the routing matrix $R \in \{-1,0,1\}^{n_q \times n_v}$ and can be activated by a network controller through a binary control-vector $v_t \in \{0,1\}^{n_v}$. Due to wireless effects, transmission over links may only succeed with probability $p_t^{[j]}$ (link j, time t). Therefore, the routing matrix is disturbed via multiplication by $M_t = \text{Bern}\left[\text{diag}_{j=1,\dots n_v}\left\{p_t^{[j]}\right\}\right]$, effectively turning some columns of R to $\mathbf{0}$ since Bern$[\cdot]$ denotes a Bernoulli trial. Note that any controller only knows R and $p_t^{[j]}$ but *not* M_t; specific modeling of the process M_t can be found in [25]. Furthermore, we usually cannot activate all links at the same time, expressed by the so-called constituency matrix $C \in \{0,1\}^{n_c \times n_v}$ and the constraint $Cv_t \leq \mathbf{1}_{n_c}$, where $\mathbf{1}_{n_c}$ is a vector of ones with dimension n_c. Finally, let $q^{[i]}$ be the queue of the entity, that the packet is destined for, then the goal of the network controller is to find a sequence of activations v_t to make $q^{[i]} = 1$.

The system so far only models transportation of one data packet and is thus referred to as a subsystem. To trace transportation of several data packets (with possibly differing origin and destination), we initialize a new copy of the subsystem each time, a communication request occurs. We can stack these subsystems together in a block diagonal manner since they will only be coupled through the constituency constraints. With slight abuse of notation we will remain with the introduced notation, however from now on are extending its meaning to include stacked variables. Note that each time, a communication request has been served, meaning that the packet has arrived at its destination, we can remove the subsystem from the stack.

Let us define a suitable control policy, that we call *Reliable Predictive Network Control* (rPNC). The goal of this policy is not only to transport data packets but also to deliver a *reliable delay forecast* over a time horizon H in each time step t, providing information of when data will arrive at its destination. To achieve this, we define a scheduled transmission to be *reliable*, if its overall transmission success probability, determined through $p_t^{[j]}$, is greater than some threshold $p_{\text{threshold}}$. Consequently, if a link has a small instantaneous success probability, it has to be activated multiple times *consecutively* to increase the overall transmission success probability. In detail, we define $r_t^{[j]} \in \mathbb{N}$, for each link $j = 1, \dots, n_v$, counting the amount of repetitions needed to ensure reliable communication for that link, when first activated in time step t. If all $p_t^{[j]}$ are governed by a discrete-time Markov chain, calculating $r_t^{[j]}$ is a straight forward task for the whole prediction horizon.

With this, a weighted graph can be constructed, in which nodes represent communication nodes, and edges represent the communication links. In this graph, $r_t^{[j]}$ are sequences (in t) of weights for each edge j, representing how many time steps one *has to* spend repeating that edge, when starting at time t, before reaching the next node. For a reliable prediction of communication delays between two plant controllers, the network controller has to determine a path between the corresponding nodes. The elapsed time for that path is given by the sum of the specific entries of the weight sequences along the way as shown in Figure 4.2. Here we identify the shortest path (in red numbers) by $\tau_0^{\text{CN}_1,\text{CN}_3} = 2+1 = 3$, so an information send from CN_1 in time-step $t = 0$ arrives at CN_2 in time-step $t = 2$ and at CN_3 in time-step $t = 3$. These delay times are used in Sec. 4 by the control system. Note that due to the definition of $r_t^{[j]}$, the real communication will probably be much faster then the predicted one. To consider the $r_t^{[j]}$ in the system evolution, we define:

$$\Gamma_t = \underset{t'=0,\dots,H-1}{\text{diag}} \left\{ \underset{j=1,\dots,n_v}{\text{diag}} \left\{ \Theta \left[t - r_{t'}^{[j]} - t' + 1 \right] \right\} \right\} \tag{4.2}$$

where diag$\{\cdot\}$ is the diagonal matrix of its arguments, and $\Theta[\cdot]$ is the Heaviside function. The matrix $\Gamma_t \in \mathbb{N}^{H \cdot n_v \times H \cdot n_v}$ (t also being the time index for the prediction) transforms $r_t^{[j]}$ into a mask function for the control-vector, being 0, when the necessary amount of repetitions for reliability has not been reached yet, and 1 otherwise.

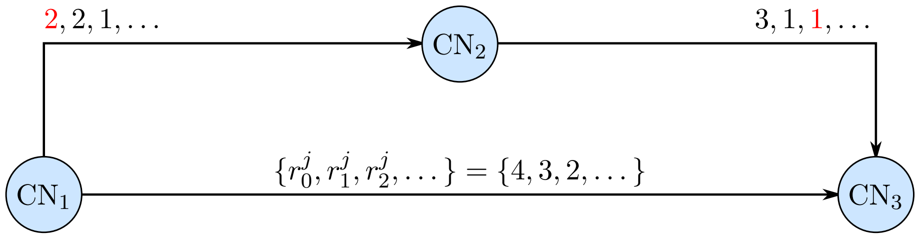


Fig. 4.2: Graph Model with weight sequences: shortest path from CN_1 to CN_3 goes through CN_2

The algorithm finds the *best* path by minimizing a cost function J_H, which assigns costs to each trajectory of queue-states up until the prediction horizon H. The cost of a queue-state can be defined in a quadratic fashion through a cost matrix Q_q, such that:

$$J_H(q_0) = \sum_{t=1}^{H} \dot{q}_t^\intercal Q_\text{q} \dot{q}_t \tag{4.3}$$

where $\dot{q}$ are states from the prediction-model that does mimic but is not identical to the system-model. We set the cost of the queue, that represents the destination entity, to a global minimum of $\dot{q}_H^\intercal Q_\mathrm{q} \dot{q}_H$. Note that for the algorithm to work, H has to be larger then the fastest weighted way between origin and destination.

For simplicity of notation, we will now assume that the current time step is $t = 0$. The predicted future queue-state is determined by the planned control decisions. We collect all planned control decisions over the horizon H in a control trajectory $\tilde{\dot{v}}_0^\intercal = \left(\dot{v}_0^\intercal,\ \dot{v}_1^\intercal, \ldots,\ \dot{v}_{H-1}^\intercal\right)$, again using $\dot{v}$ instead of v because the prediction is not performed over the actual system-model. Through their definition, $r_t^{[j]}$ and Γ_t contain the processed information of the stochastics of the communication model. Knowing Γ_t, the algorithm can therefore use a deterministic system evolution for its prediction, which will hold in a worst case sense:

$$\dot{q}_t = q_0 + \left[\mathbf{1}_H^\intercal \otimes R\right] \Gamma_t \tilde{\dot{v}}_0 \ , \qquad t = 1, \ldots, H \tag{4.4}$$

Additionally, the system has to abide by the following constraints:

- *Constituency Constraints* represent disjunct control decisions, e.g. due to physical limitations, and can be expressed via:

$$\left[I_H \otimes C\right] \tilde{\dot{v}}_0 \leq \mathbf{1}_{Hn_c} \tag{4.5}$$

- *Reliability Constraints* force the controller to consider the necessary repetitions of scheduled control decisions, in order to guarantee reliable communication. More specifically, these constraints forbid the controller to do any disjunct control decision for the appropriate amount of time steps. Define c_j^i as a row of C, which is 1 at the j-th entry, e_j as the canonical unit vector of dimension n_v, $\mathbf{0}$ as the zero vector of dimension n_v. Then for every $j = 1, \ldots, n_v$ and $t = 0, \ldots, H-1$ and every possible i, we have to construct the constraint:

$$\left(\left[\mathbf{1}_t^\intercal \otimes \mathbf{0}^\intercal\right] \quad e_i^\intercal \quad \left[\mathbf{1}^\intercal_{r_t^{[j]}-1} \otimes c_j^i\right] \quad \left[\mathbf{1}^\intercal_{H-t-r_t^{[j]}} \otimes \mathbf{0}^\intercal\right]\right) \tilde{\dot{v}}_0 \leq 1 \tag{4.6}$$

- *Consistency Constraints* guarantee, that a once communicated arrival time can *at worst* stay the same in a future optimization. Let $\delta(i)$ be the index of the destination queue of data packet i, $a(i)$ the arrival time (relative to the time of optimization) of that packet at that queue, and Δ the elapsed time between the last optimization and now. Then it must hold for each possible data packet i that:

$$-e_{\delta(i)}^\intercal \left[\mathbf{1}_H^\intercal \otimes R\right] \Gamma_{a(i)-\Delta+1}\, \tilde{\dot{v}}_0 \leq -1 + e_{\delta(i)}^\intercal q_0 \tag{4.7}$$

where this time, the dimension of e_i is n_q.

- *Processability Constraints* ensure that all queues are positive or zero at all times and forbid the routing of the same data through multiple nodes in one single time step. Defining R^- as the copy of routing matrix R in which all positive entries are set to 0, and R^+ in the reverse way, we have to implement for all $t = 1, \ldots, H$:

$$-\left[\left(\mathbf{1}_t^\intercal \otimes R^- \quad \mathbf{1}_{H-t}^\intercal \otimes \mathbf{0}\right) + \left[\mathbf{1}_H^\intercal \otimes R^+\right] \Gamma_{t-1}\right] \tilde{\dot{v}}_0 \leq q_0 \tag{4.8}$$

This completes the constrained optimization problem of minimizing (4.3). Note that the processability constraints can be relaxed to fit the physical situation. Furthermore, the consistency constraints have to be deactivated in the rare event, that reliable communication does not succeed (w.p. $1 - \phi$).

Finally, the network control policy steers the network according to the following scheme:

1. For the stacked system, obtain $\tilde{v}_0^*$ by minimizing (4.3) following the system evolution (4.4) subject to (4.5), (4.6), (4.7), and (4.8).
2. Calculate the expected delay trajectories, and communicate them to the plant controllers.
3. Apply the first part of the optimal control-vector trajectory $\tilde{v}_0^*$.
4. Recognize the new system-state and the newly requested data transmission, delete and add subsystems accordingly.
5. Repeat.

4.2.4 Control System

Consider a plant that is partitioned into a set of subsystems, each modeled as discrete-time LTI system:

$$x_{k+1}^i = A^i x_k^i + B^i u_k^i, \tag{4.9}$$

where $i \in \mathcal{N}$ indicates the subsystem, $x_k^i \in \mathbb{R}^{n_x^i}$ the local state, and $u_k^i \in \mathbb{R}^{n_u^i}$ the local control input. The state and input are subject to polytopic constraints. While the subsystem dynamics (4.9) itself are uncoupled, we consider the case that a common control goal has to be reached in the sense that a subsystem $i \in \mathcal{N}$ minimizes a cost function which depends on the state and/or input of other subsystems $j \neq i$. In order to model from where information is required, a predecessor set $\mathcal{N}_p^i$ is defined, which contains the indices of those subsystems j from which subsystem i receives information. Likewise $\mathcal{N}_f^j$ denotes the follower set containing the indices of subsystems $i \neq j$ which receive information from j.

Assumption 1. *It is assumed that the information structure established by the sets $\mathcal{N}_p^j$ and $\mathcal{N}_f^j$ is acyclic. Furthermore, we assume that the communication between two subsystems (j: sender, i: receiver) is subject to the time-varying communication delay $\tau_k^{j,i}$ introduced in Sec. 3. Additionally, we introduce $d_k^{i,j}$ as the age of the newest information i has at time step k of subsystem j.*

Given the PNC-policies introduced in Section 4.2.3 the following holds: If $i \in \mathcal{N}_f^j$, then subsystem j knows the time-delay $\tau_k^{j,i}$, and if $j \in \mathcal{N}_p^i$, then subsystem i knows the age $d_k^{i,j}$ of the next incoming information sent by j. Note that in general $\tau_k^{j,i} \neq d_{k+\tau_k^{j,i}}^{i,j}$ (since newer information with small delay may *overtake* older information).

Now, let $u_{k+l|k}^i$ denote the prediction of subsystem i of its local input at time $k+l$, calculated and sent at time k. Furthermore, let $u_{k+l|k}^{i,j}$ for $j \in \mathcal{N}_p^i$ be the prediction that subsystem i has of the inputs $u_{k+l|k}^j$ of subsystem j. Since there is a communication delay and subsystem j is, at time $k+l$, not restricted to choose u_{k+l}^j equal to the value $u_{k+l|k}^{i,j}$ as predicted and communicated earlier, subsystem i needs to consider this possible deviation. It is denoted by $\delta u_{k+l|k}^{i,j}$, and subsystem i considers this uncertainty by using a variable $\overline{u}_{k+l|k}^{i,j} = u_{k+l|k}^{i,j} + \delta u_{k+l|k}^{i,j}$ when considering the dynamics of j for evaluating its cost function.

In addition to the inputs, subsystem j communicates its current state $x^j_{k|k}$ at time k. The subsystem i knows this state exactly at time-step $k' = k + \tau^{j,i}_k$. Thus, the last state of subsystem j that is exactly known to i is $x^{i,j}_{k-d^{i,j}_k|k-d^{i,j}_k}$. With this last exactly known state and with the predicted input trajectory, subsystem i can estimate the state of j:

$$x^{i,j}_{k|k} = A^{j^{d^{i,j}_k}} x^{i,j}_{k-d^{i,j}_k|k-d^{i,j}_k} + \sum_{l=1}^{d^{i,j}_k} A^{j^{l-1}} B^j u^{i,j}_{k-l|k}. \tag{4.10}$$

Using the uncertain input and the state estimation, an augmented prediction-model can be formulated. Assuming for simplicity of notation that subsystem i has just one predecessor j, this model is:

$$\mathbf{x}^i_{k+l+1|k} = \mathbf{A}^i \mathbf{x}^i_{k+l|k} + \mathbf{B}^i u^i_{k+l|k} + \mathbf{B}^i_1 \left(u^{i,j}_{k+l|k} + \delta u^{i,j}_{k+l|k} \right), \tag{4.11}$$

with the state vector $\mathbf{x}^i_{k+l|k} = \left[x^{i^T}_{k+l|k}, x^{i,j^T}_{k+l|k} \right]^T \in \mathbb{R}^{\mathbf{n}^i_x}$, the vector of input uncertainties of the predecessor $\delta u^{i,j}_{k+l|k} \in \mathbb{R}^{n^i_u}$ where $\mathbf{n}^i_x = \sum_{j \in \mathcal{N}^i_p \cup \{i\}} n^j_x$ as well as:

$$\mathbf{A}^i = \begin{bmatrix} A^i & 0 \\ 0 & A^j \end{bmatrix}, \ \mathbf{B}^i = \begin{bmatrix} B^i \\ 0 \end{bmatrix}, \ \mathbf{B}^i_1 = \begin{bmatrix} 0 \\ B^j \end{bmatrix} \tag{4.12}$$

The stacked state vector $\mathbf{x}^i_k$ is subject to a polytopic constraint $\mathbb{X}^i = \{\mathbf{x}_k \big| \mathbf{C}^i_x \mathbf{x}_k \leq \mathbf{b}^i_x\}$.

Each subsystem can now use such a prediction-model within a robust MPC scheme to determine its own control inputs. The goal is to guarantee robustness with respect to the defined uncertainties arising from communication delay and the deviations from previously communicated trajectories. Thereto, the disturbance-feedback policy proposed in [54] can be enhanced by the predicted time-delay of the next incoming information $d^{i,j}_{k+l}$ to the following representation:

$$u^i_{k+l|k} = v^i_{k+l|k} + \sum_{r=1-d^{i,j}_k}^{l-d^{i,j}_{k+l}} K^i_{l,r|k} \delta u^{i,j}_{k+r|k}, \tag{4.13}$$

where $v^i_{k+l|k}$ is the control input in absence of uncertainties, and $K^i_{l,r|k}$ is the feedback-gain to account for the uncertainties. By formulating stacked vectors over the prediction horizon H for state, inputs, and uncertainties:

$$\begin{aligned} \tilde{\mathbf{x}}^i_k &= [\mathbf{x}^{i^T}_{k|k}, \dots, \mathbf{x}^{i^T}_{k+H|k}]^\mathsf{T}, \ \tilde{u}^i_k = [u^{i^T}_{k|k}, \dots, u^{i^T}_{k+H-1|k}]^\mathsf{T} \\ \tilde{u}^{i,j}_k &= [u^{i,j^T}_{k|k}, \dots, u^{i,j^T}_{k+H-1|k}]^\mathsf{T}, \\ \delta\tilde{u}^{i,j}_k &= [\delta u^{i,j^T}_{k+1-d^{i,j}_k|k}, \dots, \delta u^{i,j^T}_{k+H-1|k}]^\mathsf{T}, \end{aligned} \tag{4.14}$$

the following representation is obtained:

$$\tilde{\mathbf{x}}^i_k = \tilde{\mathbf{A}}^i \mathbf{x}^i_{k|k} + \tilde{\mathbf{B}}^i \tilde{u}^i_k + \tilde{\mathbf{B}}^i_1 \tilde{u}^{i,j}_k + \tilde{\mathbf{B}}^i_2 \delta\tilde{u}^{i,j}_k. \tag{4.15}$$

Equation (4.13) can be rewritten to:

$$\tilde{u}^i_k = \tilde{v}^i_k + \tilde{K}^i_k \delta\tilde{u}^{i,j}_k, \tag{4.16}$$

with $\tilde{v}_k^i \in \mathbb{R}^{H\,n_u^i}$ and the uncertainty feedback matrix $\tilde{K}_k^i \in \mathbb{R}^{H\,n_u^i \times n_{1,k}^i}$, where $n_{1,k}^i\left(d_k^{i,j}\right)$ represents the time varying length of $\delta\tilde{u}_k^{i,j}$. Similarly as in [56], the lower triangular block matrix $\tilde{K}_k$ can be enhanced with the prediction of $\tilde{d}_k^{i,j} = [d_k^{i,j}, \ldots, d_{k+N-1}^{i,j}]$.

Let constraints for the stacked vectors of state, input, and uncertainties over the prediction horizon be denoted by:

$$\tilde{\mathbf{x}}_k \in \tilde{\mathbb{X}}^i, \quad \tilde{u}_k \in \tilde{\mathbb{U}}_k^i, \quad \delta\tilde{u}_k \in \delta\tilde{\mathbb{U}}_k^i. \tag{4.17}$$

When introducing auxiliary matrices $\mathbf{F}_1$ to $\mathbf{F}_5$ according to [54]), the admissible set of input sequences can be defined to:

$$\Pi_k^i\left(\mathbf{x}_0\right) = \left\{ \left(\tilde{K}_k^i,\, \tilde{v}_k^i\right) \,\middle|\, \begin{array}{l} \mathbf{x}_{k|k}^i = \mathbf{x}_0,\ \exists Z_k^i \geq 0 : \\ Z_k^i \tilde{\mathbf{C}}_\delta^i = \mathbf{F}_2^i \tilde{K}_k^i + \mathbf{F}_4^i \\ \mathbf{F}_2^i \tilde{v}_k^i + Z_k^i \tilde{b}_{\delta_k}^i \leq \ldots \\ \quad \mathbf{F}_5^i - \mathbf{F}_1^i \mathbf{x}_{k|k}^i - \mathbf{F}_3^i \tilde{u}_k^{i,j} \end{array} \right\}, \tag{4.18}$$

with slack variables $Z_k^i = \left[Z_{x_{k|k}}^{iT},\, Z_{u_{k|k}}^{iT}\right]^\mathsf{T}$, allowing us to characterize the set of possible reactions to uncertainties in (4.16) by:

$$\Delta\tilde{\mathbb{U}}_k^i = \left\{ \Delta\tilde{u}_k^i \in \mathbb{R}^{H\,n_u^i} \,\middle|\, \tilde{C}_u^i\, \Delta\tilde{u}_k^i \leq Z_{u_{k|k}}^i \tilde{b}_{\delta_k}^i = \tilde{b}_{\Delta_k}^i \right\},$$

with $\Delta\tilde{u}_k^i = \tilde{K}_k^i \delta\tilde{u}_k^{i,j}$ and $\tilde{b}_{\Delta_k}^i = \left[b_{\Delta_{k|k}}^{iT}, \ldots, b_{\Delta_{k+H-1|k}}^{iT}\right]^\mathsf{T}$. This set represents also the set of uncertainties to be communicated to a successor of subsystem i. With respect to terminal sets and terminal control laws to establish recursive feasibility, we refer the reader to the solution proposed in [54].

For each subsystem, a two-stage optimization problem is now to be solved in any k: The first stage minimizes the cost function formulated to be quadratic in the augmented state and control inputs (thus depending on the planned and communicated inputs of the predecessors), but without considering the uncertainties:

$$\begin{aligned} J_1^i &= \left\|\mathbf{x}_{k+H|k}^i\right\|_{Q_T^i}^2 + \sum_{l=0}^{H-1} \left\|\mathbf{x}_{k+l|k}^i\right\|_{Q_x^i}^2 + \left\|\left[u_{k+l|k}^{iT} \quad \tilde{u}_{k+l|k}^{i,jT}\right]\right\|_{Q_u^i}^2 \\ V_1^i\left(\mathbf{x}_0^i\right) &:= \min_{\tilde{v}_k^i,\, \tilde{K}_k^i} J_1^i \\ \text{s.t.: } & \mathbf{x}_{k|k}^i = \mathbf{x}_0^i, \quad \left(\tilde{K}_k^i,\, \tilde{v}_k^i\right) \in \Pi_k^i\left(\mathbf{x}_0^i\right), (4.17), (4.18), \\ & \quad (4.11), (4.13) \text{ with } \delta u_{k+l|k}^{i,j} = 0\ \forall l \in \{0, \ldots, H-1\}. \end{aligned} \tag{4.19}$$

The entries in $\tilde{K}_k^i$ are not uniquely defined for $V_1^i\left(\mathbf{x}_0^i\right)$, and since $\tilde{K}_k^i$ affects the set $\Delta\tilde{\mathbb{U}}_k^i$, $\tilde{K}_k^i$ is optimized in a second stage, still satisfying (4.18). Using two uncertainty sets $\delta\tilde{\mathbb{U}}_k^i = \left[\delta\mathbb{U}_{k+1-d_k^{i,j}|k}^i, \ldots, \delta\mathbb{U}_{k+H-1|k}^i\right]$ and $\Delta\tilde{\mathbb{U}}_k^i = \left[\Delta\mathbb{U}_{k|k}^i, \ldots, \Delta\mathbb{U}_{k+H-1|k}^i\right]$, this cost function is defined as:

$$J_2^i = \sum_{l=1}^{H-d_k^{i,j}-1} f_1\left(\delta\mathbb{U}_{k+l|k}^i, \Delta\mathbb{U}_{k+l|k}^i\right) + \sum_{l=0}^{\tau_k^i} f_2\left(\Delta\mathbb{U}_{k+l|k}^i\right),$$

where $\tau_k^i = \max_{j \in \mathcal{N}_f^i}\left(\tau_k^{i,j}\right)$ is the maximum delay of outgoing information. In the cost function, f_1 is used to balance the incoming and outgoing uncertainty sets $\left(\delta\tilde{\mathbb{U}}_k^i, \Delta\tilde{\mathbb{U}}_k^i\right)$, i.e. the main idea is to preserve for $k+l$ the same flexibility for adapting $u_{k+l|k}^i$ with $\Delta u_{k+l|k}^i \in \Delta\mathbb{U}_{k+l|k}^i$ as the predecessor has predicted for its control input through $\delta u_{k+l|k}^j \in \delta\mathbb{U}_{k+l|k}^j$. This balancing avoids an blow-up of uncertainty sets by propagating the uncertainties through the interconnected graph. The term f_2, on the other hand, is used to tighten the uncertainty set $\Delta\tilde{\mathbb{U}}_k^i$ for the next τ_k^i time steps required to pass information to the follower.

The optimization of the second stage is now:

$$V_2^i\left(\mathbf{x}_0^i, \tilde{v}_k^i\right) = \min_{\tilde{K}_k^i} J_2^i \quad \text{s.t.:} \ \left(\tilde{K}_k^i\right) \in \Pi_k^i\left(\mathbf{x}_0^i, \tilde{v}_k^i\right). \tag{4.20}$$

Since the control input trajectory $\tilde{v}_k^i$ is already computed in (4.19), this is preset in (4.20) to the set of admissible input sequences Π_k^i. Thus, (4.20) just adapts the values of $\tilde{K}_k^i$ and, consequently, the behavior of the controlled system in the next time steps (but not in the current time step k).

4.2.5 Simulation Example

To illustrate the methods and the resulting gain in control performance, the approach is applied to the platooning example sketched in Figure 4.3. We assume that CPS 1 follows an uncertain trajectory, while CPS 2 and CPS 3 aim at maintaining the position and speed of the predecessor, while satisfying acceleration constraints.

The communication network is organized as a cellular network, so every incoming

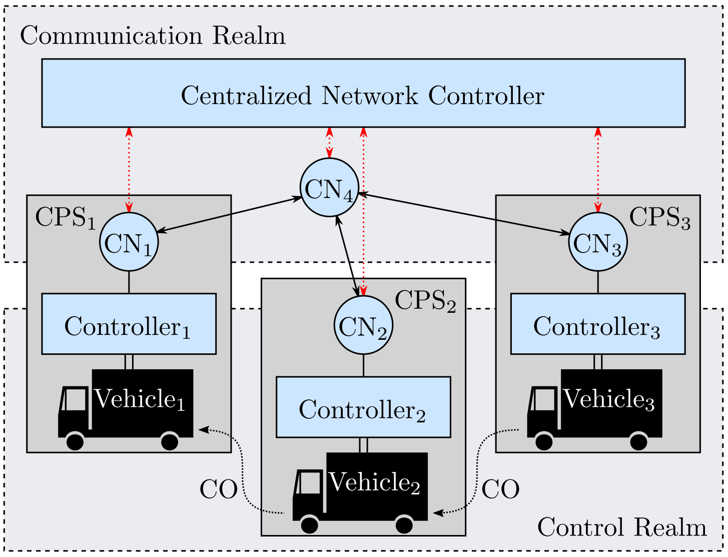


Fig. 4.3: Platooning example of a cyber-physical network. CN: Communication Node, CO: Control objective.

information is send to the base station CN$_4$, and then transmitted to a selected receiver. The selection and activation of communication links is controlled by the network controller with the introduced PNC-policies. For the sake of illustration, we restrict the delay $\tau_k^{2,3} = 1$ to a constant value. In contrast, $\tau_k^{1,2}$ is time-varying, predictable by PNC-policies, and thus communicated to CPS 1 and CPS 2. Local dynamics and input constraints of all CPS are assumed to be identical:

$$x_{k+1}^i = \begin{bmatrix} 1 & 0.3 \\ 0 & 1 \end{bmatrix} x_k^i + \begin{bmatrix} 0.045 \\ 0.3 \end{bmatrix} u_k^i, \quad u_k^i \in [-4,\ 4] \tag{4.21}$$

with $x_k^i = [x_{1,k},\ x_{2,k}]^\mathsf{T}$, where $x_{1,k}$ is the position, $x_{2,k}$ the velocity, and u_k^i the acceleration. Hence, the augmented prediction-model is given by $\mathbf{x}_k^i = \left[x_k^{i^T}, x_k^{i-1^T} \right]^\mathsf{T}, i \in \{2, 3\}$, and the augmented state constraints are specified as $\left(x_k^i - x_k^{i-1}\right) \in [10,\ 10] \times [10,\ 10]$. The cost function $J_2^i, i \in \mathcal{N}$ contains $f_1\left(\delta\mathbb{U}_{k+l|k}^i, \Delta\mathbb{U}_{k+l|k}^i\right) = \left\| b_{\delta_{k+l|k}}^i - b_{\Delta_{k+l|k}}^i \right\|^2$, and $f_2\left(\Delta\mathbb{U}_{k+l|k}^i\right) = \left\| b_{\Delta_{k+l|k}}^i \right\|^2$. For a prediction horizon of $H = 5$, it is assumed that the PNC-policies guarantee $\tau_k^{1,2} \leq \bar{\tau}^{1,2} = 4\ \forall k$, hence $d_k^{2,1} \leq \bar{\tau}^{1,2}\ \forall k$ holds. In the following, the simulation results of the presented methods (new) are summarized and compared to the worst case delay method (wc d) from [54]. The simulated scenario is the following: at $k = 0$, the position, velocity, and acceleration of all vehicles are initialized to zero. At $k = 10$, the input reference trajectory of CPS 1 steps unpredicted to 1.5, thus CPS 1 starts accelerating while maintaining the previously communicated uncertainty sets. At $k = 40$, the reference steps back to zero, but this time predicted before. For the sake of comparison, the (in general time-varying) communication delay $\tau_{10}^{1,2}$ is fixed at $k = 10$ to the maximum value, so the step of CPS 1 is known to CPS 2 with $d_{14}^{2,1} = 4$. Figure 4.4 shows the control input of CPS 1 (a) and CPS 2 (b). Since the new methodology already tightens the uncertainty set $\Delta\tilde{\mathbb{U}}_k^1$ at $k = 9$ while knowing the output delay, CPS 1 cannot react that agile at $k = 10$ and $k = 11$. Once the step of the control input is communicated, CPS 2 can react much more aggressive in $k = 14$ and $k = 15$ due to the smaller uncertainty set communicated by CPS 1 in $k = 9$. The slower behavior of the input of CPS 1 leads to a worse control result related to the reference, as plotted in Figure 4.4, bottom left, (while CPS 1 follows the reference perfectly in the worst case delay simulation). This degradation is acceptable, and even intended, since CPS 2 can react much faster, as shown in Figure 4.4, bottom right.

Summing up all quadratic deviations between control result and reference trajectory according to the weights in J_1, the performance can be compared for both approaches, see Table 4.1. The little worse result for CPS 1 (less than 2) has to be contrasted to drastic improvements for CPS 2 (more than 130 points) and even for CPS 3. Remembering that the communication delay $\tau_k^{2,3} \equiv 1$, the better performance measure for CPS 3 is most significant. Since there is no difference between a worst case delay and a predicted delay of constant one, the improved control result is an effect of the second optimization stage.

Table 4.1: Performance measure of platoon

	CPS 1	CPS 2	CPS 3	total
worst case delay	0	275.55	288.37	563.92
new methodology	1.52	141.56	248.33	391.41

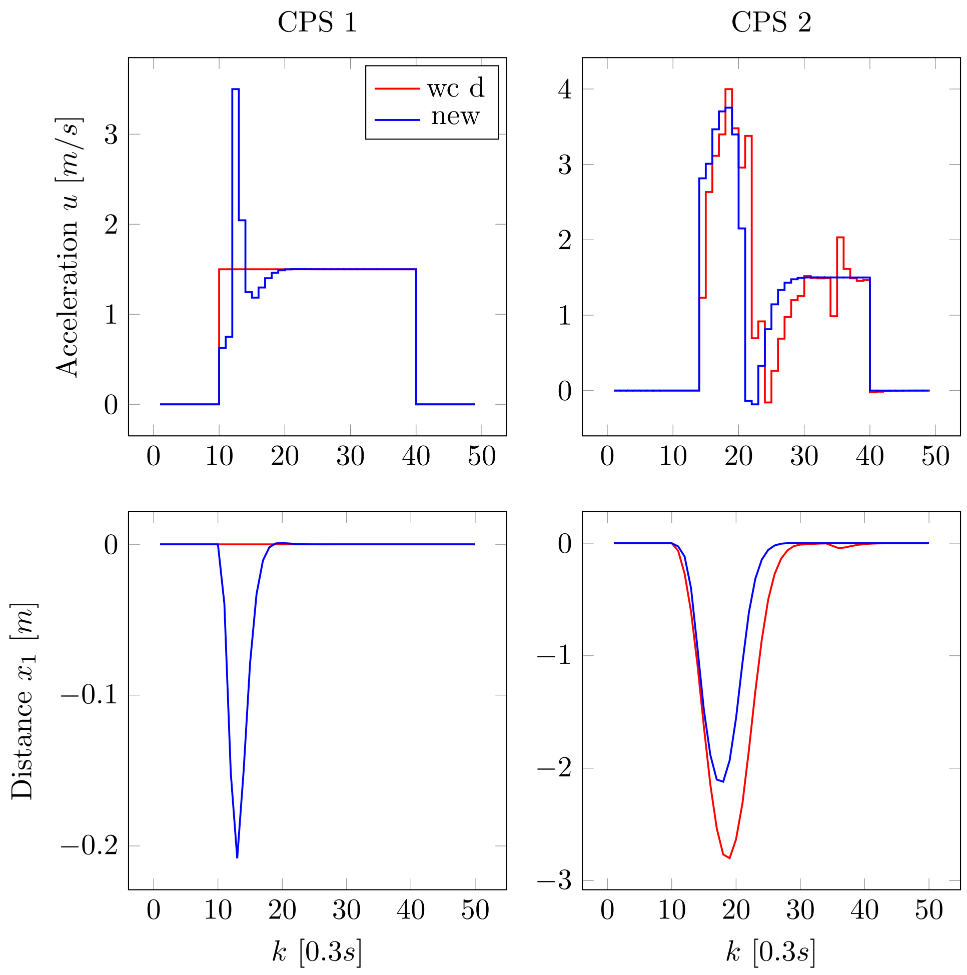


Fig. 4.4: Control input and distance to reference

4.2.6 Conclusion

We have proposed a new theory for combining predictive control of communication networks with distributed MPC for CPS. If a describing model of the communication network exists, it is advisable to control the communication network with the presented RPNC-algorithm. Reducing the overall buffer size, it produces a prediction of future transmitting delays, which is usable by the associated distributed control system in two ways: Firstly, incorporating the predicted age of future incoming data packages can increase the degree of freedom of computing the robust control invariant sets. Secondly, the prediction of sending delay can be used to optimize the control invariant set for the following CPS. In comparison to a worst case communication delay, our methodology improves the overall control result of a platooning example by over 30%.

4.3 Concluding Remarks

Due to a page limitation, the paper lags an illustrative example for the explanation of the network control algorithm. This shall be corrected in this section, starting with a few notes on the general idea behind the algorithm:

1. The reliability concept on which the algorithm is constructed is concerned

only with reliable transmission via a single link. E.g. defining $p_{\text{threshold}} = 0.99$ only makes sure that transmission is repeated as often as is necessary to guarantee that each link individually will have successfully transmitted the packet with probability beyond 99%. However, if there are 5 links that have to be traversed one after another in order to reach the final destination, then the overall probability of the packet reaching the final destination is only guaranteed to be higher than $0.99^5 \approx 0.95$. If this probability is to be higher than 99%, one would have to set the threshold to $p_{\text{threshold}} = 0.99^{\frac{1}{5}} \approx 0.998$.

2. The matrix Γ_t is, of course, reevaluated in every time step, as are the $r_t^{[j]}$. Being a constant ingredient in all our papers, we assume that the underlying stochastic process that governs the transmission success probabilities of the links is (or at least can be approximated by) a discrete-time Markov chain. Hence, every new time-step brings with it the additional information of the current Markov-state of this DTMC. This allows for a better estimation of future transmission success probabilities and therefore better estimations for the $r_t^{[j]}$.

3. The presented algorithm features at its core a minimization. Discussing its objective, the paper states that "We set the cost of the queue, that represents the destination entity, to a global minimum of $q_H^{\intercal} Q_{\text{q}} q_H$". This is a insufficient description that needs to be specified. First, we implicitly assume that Q_{q} is a diagonal matrix because there is no apparent reason to assign any cost to the product of $q^{[i]}$ and $q^{[j]}$ if $i \neq j$. Second, the diagonal element of Q_{q} that combines $q^{[d]}$ with itself (where d is the index of the destination queue) is set to a very large negative value compared to all other diagonal elements. This way, any minimization is strongly motivated to assign large values to $q^{[d]}$ which can only happen if the system steers packets towards this queue.

Next, we explore the algorithm using the minimal example, depicted in Figure 4.5. The communication request initiates a packet in $q^{[1]}$ that is supposed to be delivered to $q^{[3]}$. The routing matrix and constituency constraints can be written as

$$R = \begin{pmatrix} -1 & 0 \\ +1 & -1 \\ 0 & +1 \end{pmatrix} \quad , \quad \underbrace{\begin{pmatrix} 1 & 1 \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} v^{[1]} \\ v^{[2]} \end{pmatrix}}_{v} \leq 1 \tag{4.22}$$

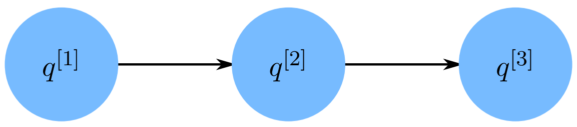


Fig. 4.5: Simple example of 3 queues on which to illustrate the principle of the algorithm.

Based on a given description of the transmission failure probabilities of the links, we assume that the r-values have been calculated to form the following sequence of vectors:

$$r_0, r_1, r_2, \dots = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \dots \tag{4.23}$$

Note that these values are calculated in the current time-step $t = 0$ and describe how many times a link must be activated in order to ensure reliable communication. The Γ-matrices then take the form

$$\Gamma_{-1} = \operatorname{diag}\begin{Bmatrix}0\\0\\0\\0\\0\\0\end{Bmatrix}, \Gamma_0 = \operatorname{diag}\begin{Bmatrix}0\\0\\0\\0\\0\\0\end{Bmatrix}, \Gamma_1 = \operatorname{diag}\begin{Bmatrix}1\\1\\1\\1\\0\\0\end{Bmatrix}, \Gamma_2 = \operatorname{diag}\begin{Bmatrix}1\\1\\1\\1\\1\\1\end{Bmatrix} \tag{4.24}$$

The entries in Γ_t act like a mask on the routing matrix. Since $r_0 = (2, 2)^\intercal$, communication is reliable only when two consecutive activations are initiated, starting in the first time-step $t = 0$. And because $r_1 = r_2 = (1, 1)^\intercal$, a transmission is immediately reliable when starting in time steps $t = 1$ or $t = 2$. I.e. earliest in time step $t = 3$ a packet is allowed to appear in $q^{[3]}$. Accordingly, the evolution of the prediction-model becomes:

$$\begin{aligned}
\dot{q}_1 &= \begin{pmatrix}1\\0\\0\end{pmatrix} + \left(\begin{array}{c|c|c} \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right)\tilde{\dot{v}}_0 \\
\dot{q}_2 &= \begin{pmatrix}1\\0\\0\end{pmatrix} + \left(\begin{array}{cc|cc|c} -1 & 0 & -1 & 0 & \\ +1 & -1 & +1 & -1 & \mathbf{0} \\ 0 & +1 & 0 & +1 & \end{array}\right)\tilde{\dot{v}}_0 \\
\dot{q}_3 &= \begin{pmatrix}1\\0\\0\end{pmatrix} + \left(\begin{array}{cc|cc|cc} -1 & 0 & -1 & 0 & -1 & 0 \\ +1 & -1 & +1 & -1 & +1 & -1 \\ 0 & +1 & 0 & +1 & 0 & +1 \end{array}\right)\tilde{\dot{v}}_0
\end{aligned} \tag{4.25}$$

In order to force the algorithm to activate a link as many times as necessary, the reliability constraints become

$$\begin{pmatrix}1&0&1&1&0&0\\0&1&1&1&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{pmatrix}\begin{pmatrix}\dot{v}_0^{[1]}\\\dot{v}_0^{[2]}\\\dot{v}_1^{[1]}\\\dot{v}_1^{[2]}\\\dot{v}_2^{[1]}\\\dot{v}_2^{[2]}\end{pmatrix} \leq \mathbf{1} \tag{4.26}$$

Notice how activating the first link in time step $t = 0$ ($\dot{v}_0^{[1]}$) does impede activation of the same link in time step $t = 1$ ($\dot{v}_1^{[1]}$). At a first glance, this seems wrong since we wanted the link to be activated twice in a row in order to meet our reliability goal. However, activating the link twice in a row would decrease the source queue and increase the destination queue by 2 packets, which would violate the processability (positiveness) constraints. Therefore, the constraints (4.6) are constructed in such a way that activation of the same link, or any interfering link, is blocked entirely in the subsequent time steps (for as long as the reliability goal specifies). Remember that by virtue of the Model-Predictive Control paradigms, the actual system evolution can indeed activate the link twice (if necessary). It is merely the prediction in which a second activation is replaced by a quasi-idle control decision.

The processability constraints become

$$-\left[\begin{pmatrix} R^- & 0 & 0 \\ R^- & R^- & 0 \\ R^- & R^- & R^- \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ R^+ & R^+ & 0 \end{pmatrix}\right] \tilde{\tilde{v}}_0 \leq \begin{pmatrix} q_0 \\ q_0 \\ q_0 \end{pmatrix} \tag{4.27}$$

For the first 2 time-steps ($t = 0$ and $t = 1$), only the draining effect of $\tilde{\tilde{v}}_0$ is considered and hence, only the source queue can be drained (since the system was initialized with exactly one packet in the source queue). It is only until time step $t = 2$ that the filling effect is considered (through the positive entries in R which are added by R^+) and thus only in this time step the constraints do allow for draining of other queues (here, $q^{[2]}$).

In this scenario, it is also possible for the controller to save resources and schedule the first transmission for time-step $t = 1$ (instead of $t = 0$), since then it does not need to repeat the activation a second time (as was mandatory when starting the first transmission in time-step $t = 0$ in order to obtain reliability). If $r_1^{[0]}$ would be 2 instead of 1, the controller would not be motivated to do so, since then, activation of $v_2^{[2]}$ would be impossible due to the reliability constraints.

Paper 5

Linear Reliable Prediction

Full Paper Title A Linear Algorithm for Reliable Predictive Network Control

Publisher IEEE Global Communications Conference (GLOBECOM), 2018

Authors Richard Schoeffauer, Gerhard Wunder

Acknowledgment This research is partially supported by the EU H2020-ICT2016-2 project ONE5G and the DFG Priority Programme 1914 Cyber-Physical Networking (CPN). The views expressed in this paper are those of the authors and do not necessarily represent the project views.

Abstract This paper introduces a predictive and reliable control approach for network scheduling and routing for arbitrary topologies in discrete-time, packet level formulation. We assume existence of a Markov chain, governing the behavior of the network links, so that our controller can optimize its decisions over a finite time horizon. Furthermore, we define a notion of reliability in order to facilitate meaningful forecasts of individual packet delays which can be used by the recipient network agents. We make a point to formulate the optimization problem as a linear program for the purpose of its fast execution in practice. With extended simulations, we demonstrate the gains in performance over the well known MaxWeight policy.

5.1 Preliminary Remarks

This Paper presents yet another approach to predicting package arrival in a queueing network in a reliable manner. However, the final problem to optimize in this paper is a binary linear program, making it much easier to solve compared to the binary quadratic program from the previous paper. Still, the fundamental method to incorporate reliability remains the repetition of transmissions with low success probability.

5.2 Paper Body

5.2.1 Introduction

The fifth generation of mobile communication, 5G, aims at not only enabling communication between billions of people around the globe but also connecting billions of devices. In this context, many boundaries are currently tackled by research, such as increasing data rates, providing security and many more.

This paper is dedicated to enhance performance of networked devices through a *predictive* scheduling and routing of data packets through the network. Specifically, the here presented network control policy enhances the performance of interconnected *robust model-predictive controllers* (RMPCs). The policy does not only schedule and route data, but as a novel feature, also predicts their time of arrival at their corresponding destination. In other words it predicts the communication delays. Signaling these communication delays ahead of the arrival of the actual data to the corresponding RMPCs facilitates them to enhance their control performance, as was first shown in [57].

As a reference network control policy, we will use the well known MaxWeight or MaxPressure policy, (from now on written as MW), first introduced by [2]. In the last two decades, MW and other network control strategies were investigated intensively, e.g. in [5]. However, focus has yet always remained at lowering overall delay while maintaining the property of maximum throughput, which makes MW such a good policy in the first place [4] [58]. And though prediction has been successfully used to improve overall delay in broadcast scenarios [8], for the best of our knowledge, trying to predict individual packet delays is a novel idea.

Another kindred topic is the so-called delay-constrained routing and scheduling. While originating from area of wired communication [59], it has also been investigated for the wireless case [60]. However, results are yet limited to rather mathematical statements with limited use for practice.

In this paper we represent a fast algorithm to schedule and route information through a network and at the same time provide forecasts of specific delay times in a *reliable* fashion. We build on the ideas of [25] where we described a first approach to yield reliable delay forecasts. However the algorithm that was presented was computationally expensive (due to its quadratic nature) and had worse performance in the achieved delay times (caused by a strictly repetitive activation of links). The algorithm designed in this paper, will eliminate these shortcomings.

5.2.2 System model

We make use of the standard queueing model, which is time-discrete, integer valued and offers binary controls. This is the appropriate choice for packet level modeling. Each of the $\bar{n}$ agents in the network may hold multiple (data-) packets at a given time-step t, which are to be transmitted to other agents. Those packets are lined up in so-called queues $\bar{q}^{[i]}, i = 1, \dots \bar{n}$. E.g. $\bar{q}^{[i]} = 4$ represents 4 packets, waiting in queue i (located at agent i). The vector of all queues will be denoted as $\bar{q} \in \mathbb{N}^{\bar{n}}$. With this model, sending packets from one agent to another is represented by decreasing and increasing queues at the corresponding agents. De- and increasing can be done by a vector $r^{[j]} \in \{-1, 0 - 1\}^{\bar{n}}$, $j = 1, \dots \bar{m}$, which is called a link; the matrix of all links is called the routing matrix $R \in \{-1, 0, 1\}^{\bar{n} \times \bar{m}}$. In each time-step, we can choose to activate a link through a binary control-vector $\bar{u} \in \{0, 1\}^{\bar{m}}$, so that the system evolves like

$$\bar{q}_{t+1} = \bar{q}_t + \bar{R}\bar{M}_t\bar{v}_t + \bar{a}_t \ , \quad \text{s.t.} \quad \bar{C}\bar{v}_t \leq \mathbf{1} \ , \quad \bar{q}_{t+1} \geq 0 \tag{5.1}$$

where $\mathbf{1}$ is a vector of ones with appropriate dimension. The arrival $\bar{a}_t \in \mathbb{N}^{\bar{n}}$ expresses an influx of information to the system and is usually of stochastic nature. The constituency matrix $\bar{C}$ prohibits to activate all controls simultaneously, and naturally queues can only hold a positive number of packets, giving rise to the positiveness constraint. Note, that it is a key feature of *wireless* networks to have a (stochastically) time dependent *current* topology, expressed by the multiplication $\bar{R} \cdot \bar{M}_t$. In each time-step, the diagonal matrix $\bar{M}_t$ takes a random selection of the columns from $\bar{R}$ and forces them to become $\mathbf{0}$ columns in the product. The selection process is given through transmission success probabilities $\bar{p}^{[i]} \in [0, 1]$ for each link i and a Bernoulli trial $\text{Bern}[\cdot]$ (where $\text{Bern}[\xi] = 1$ w.p. ξ and 0 otherwise) performing on it. These probabilities are collected in a diagonal matrix $\bar{W} = \underset{i=1\dots m}{\text{diag}} \left\{\bar{p}^{[i]}\right\}$, so that we can express the matrix $\bar{M}_t$ as $\bar{M}_t = \text{Bern}\left[\bar{W}\right]$. Notice that a controller only knows $\bar{R}$ and $\bar{W}$ but not $\bar{M}_t$, thus not every activation (scheduled transmission) will succeed. The problem lies in finding the best suited control to steer the data to its destination, though not fully knowing the outcome of the Bernoulli trial.

In order to use meaningful predictions of future behavior, we enhance this standard model by defining a whole set of probability matrices $\bar{W}^i \in \mathcal{W}$ instead of only one (as already done in [25]). In each time-step, the system uses one $\bar{W}^i$, according to a discrete-time Markov chain, that evolves on the index set $\mathcal{I}(\mathcal{W})$ as $s_t = \mathbb{M}\left(\mathcal{I}(\mathcal{W}), P, s_0\right)$, P being the transition matrix and s_0 the initial Markov-state. Hence we get $\bar{M}_t = \text{Bern}\left[\bar{W}^{s_t}\right]$. We assume that the controller has full knowledge of $\mathcal{W}$, P, and s_t (the current Markov-state), i.e. knows the expected transmission success probabilities of all links and all future times. The network controller may be required to measure these parameters before being able to control the network. In this transient state, the network may be controlled by the MW policy.

5.2.3 Reliable Predictive Network Control

To predict packet transmissions in the system-model, the network controller internally uses a slightly different model to predict the flow of the packets. This so-called *prediction-model* is shown in Figure 5.1 and will be described in this section. Two major circumstances give rise to the prediction-model:

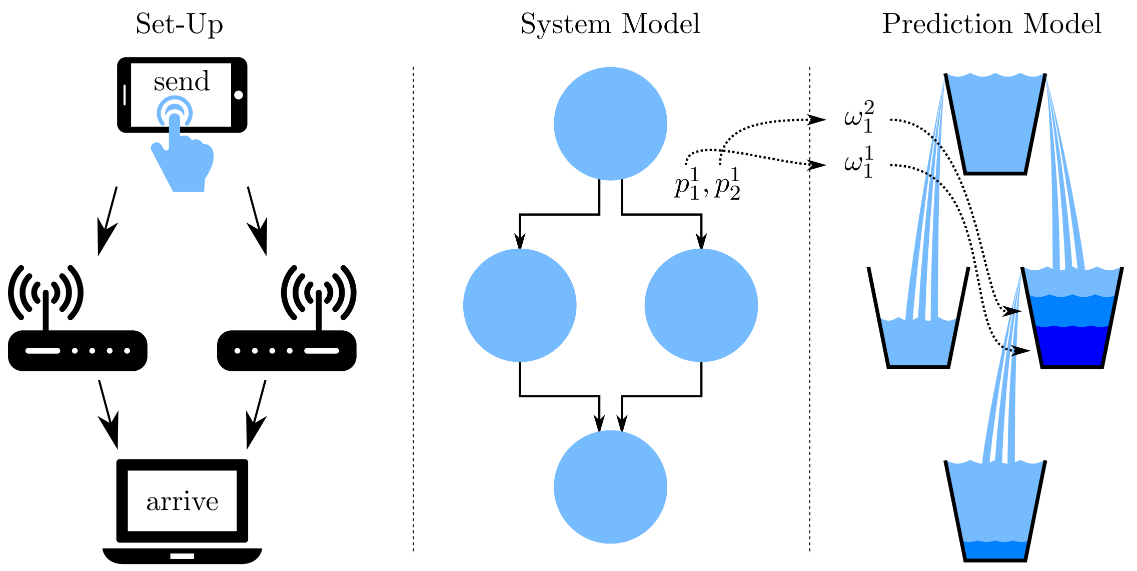


Fig. 5.1: Compared to the system-model, the prediction-model uses continuous values for the states and can be visualized as an arrangement of buckets being filled by dripping water.

1. Two packets, simultaneously residing at the same agent, may still be intended for different destination agents. Hence, in order to distinguish different packets mathematically, each one of them has be cast with its own copy of the system-model (5.1). These copies will be called subsystems. Suppose that z is the number of subsystems currently in use (i.e. z packets are currently present in the network), then we define

$$\begin{aligned} R &:= I_z \otimes \bar{R} \\ W^i &:= I_z \otimes \bar{W}^i, \\ M_t &:= I_z \otimes \bar{M}_t \end{aligned} \qquad u_t := \begin{pmatrix} \bar{u}_t{}^{(1)} \\ \vdots \\ \bar{u}_t{}^{(z)} \end{pmatrix}, \quad q_t := \begin{pmatrix} \bar{q}_t{}^{(1)} \\ \vdots \\ \bar{q}_t^{(z)} \end{pmatrix} \tag{5.2}$$

where $(\cdot)^{(i)}$ corresponds to the i-th subsystem, I denotes the identity matrix, and $n = \bar{n} \cdot z$ and $m = \bar{m} \cdot z$ denote the dimensions of the stacked vectors. Note that C might be constructed in a different way depending on the scenario. The system evolution thus becomes

$$q_{t+1} = q_t + RM_t u_t \tag{5.3}$$

Here, we can ignore the arrival $\bar{a}_t$ since any new packet arriving at the system will immediately lead to another subsystem being cast and stacked on top the current prediction-model. E.g. if agent i signals the initialization of a new packet, then subsystem $(z+1)$ will be cast and the packet will be represented by initializing $\bar{q}_t^{(z+1)}$ with a 1 at the i-th component. In the same way, a subsystem can be erased from the stack, once the agent, the packet was intended for, signals (to the controller) its successful delivery.

2. The agents, here RMPCs, are only able to use the forecasts of packet-specific communication delays, if these forecasts are reliable (so that the RMPCs can still guarantee robustness). Therefore, our policy will predict the flow through the network with explicit consideration of possible transmission failures. Specifically, in its prediction for future activations, the algorithm will

repeat activating a link, until the accumulative expected transmission failure probability falls beneath a *single-transmission failure-probability threshold* τ. We use dotted variables $\dot{q}_t$ and $\dot{v}_t$ to denote queue-states and control-vectors in the prediction-model. Let f_t^i be the *expected* transmission failure probability for time-step t (the current time-step being 0) and link i, given by

$$\begin{pmatrix} f_t^{[1]} & & \\ & \ddots & \\ & & f_t^{[m]} \end{pmatrix}_t = I_m - \left[\sigma_0 P^t \otimes I_m\right] \begin{pmatrix} W^1 \\ \vdots \\ W^{|\mathcal{W}|} \end{pmatrix} \tag{5.4}$$

where σ_t is the distribution of s_t such that $\sigma_0 P^t$ (here, t is an exponent) is the distribution of s_t, predicted at $t = 0$. Then we define that a packet is predicted to be reliably transmitted over a link i (in the prediction), only if

$$\prod_{t=0}^{H-1} {f_t^{[i]}}^{\dot{v}_t^{[i]}} \leq \tau \qquad \Longrightarrow \qquad \sum_{t=0}^{H-1} \dot{v}_t^{[i]} \log_\tau f_t^{[i]} \geq 1 \tag{5.5}$$

where $\dot{v}_t^{[i]}$ is the corresponding control variable for this link in the prediction-model and H is the prediction horizon.

Now we return to the network controller. It is implemented as an MPC, meaning that in *each* time-step, the controller minimizes a cost function (influenced by the prediction-model) to yield a control trajectory $\tilde{\dot{v}}_0^\intercal = \left(\dot{v}_0^\intercal, \dots \dot{v}_{H-1}^\intercal\right)$ over a horizon H but then only applies the first component to the system, such that $v_0 = \dot{v}_0$. (Here, the current time step is set to 0 for ease of notation.) Usually, MPC objective functions are quadratic in nature, leading to semi definite programs. Since network control has to happen very fast (depending on the granularity of the data), a main contribution in this paper is to devise an algorithm that is specifically designed to be a binary linear program which is solvable in polynomial time [61].

The intuition behind the algorithm equals a waterfall, always filling the queues in direct vicinity of already filled ones. As a first step we introduce the reliability (5.5) as a constraint. Let

$$\omega_t^{[i]} = \min\left\{ \log_\tau f_t^{[i]} \quad , \quad 1 \right\} \tag{5.6}$$

then we can define

$$\Omega^C = \left(\operatorname*{diag}_i \left\{\omega_0^{[i]}\right\} \quad \dots \quad \operatorname*{diag}_i \left\{\omega_{H-1}^{[i]}\right\} \right) \in \mathbb{R}^{m \times mH} \tag{5.7}$$

Forcing $\Omega^C \tilde{v}_0 \geq \mathbf{1}$ will make $\tilde{v}_0$ guarantee reliable activations and hence reliable forecasts as described earlier. However, applying this to our system evolution so far could contradict the positiveness constraint on q. E.g. having three scheduled activations $v^{[i]}$ (over different time-steps during the horizon H) of the link i would result in a negative queue-state of $1-3=-2$ at the link-origin queue and suggest the presence of $0+3=3$ data packets at the link-destination queue. To compensate, we bring two changes to the prediction-model. First, we change the $r^{[i]}$ (links) to not *de*crease any queue-states at all. Second, we multiply the $r^{[i]}$ with their corresponding $\omega_t^{[i]}$ values, resulting in the queue vector being real valued ($\dot{q}_t \in \mathbb{R}^n$)

in th prediction-model. Incorporating these changes into (5.3) while simultaneously considering the evolution for the whole prediction horizon yields

$$\begin{pmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_H \end{pmatrix} = \begin{pmatrix} q_0 \\ \vdots \\ q_0 \end{pmatrix} + \begin{pmatrix} R^+ & & \\ \vdots & \ddots & \\ R^+ & \dots & R^+ \end{pmatrix} \Omega^E \tilde{\dot{v}}_0 \tag{5.8}$$

with the two definitions

$$R^{+[i,j]} = \max \left\{ \ 0 \ \ , \ \ R^{[i,j]} \ \right\} \tag{5.9}$$

and

$$\Omega^E = \operatorname*{diag}_{t=0\dots H-1} \left\{ \operatorname*{diag}_{i=1\dots m} \left\{ \omega_t^{[i]} \right\} \right\} \tag{5.10}$$

Note that ignoring to decrease queues is only viable because we are using separate subsystems for each single packet and therefore need only to consider the propagation of the packet but not what happens to queues that have already been passed by it. This way, we also avoid further constraints for the positiveness of the queues. Furthermore, using $\omega_t^{[i]}$ as weights means that a packet is predicted to have successfully been transmitted over a link, if the link-destination queue is filled exactly to or beyond 1. However we are still missing two main ingredients for the prediction-model to work: on the one hand, queues filled just beyond 1 are not supposed to be filled any further. On the other, only those links can be activated, whose link-origin queue has been filled exactly to or beyond 1.

By the virtue of $\dot{v}_t^{[i]}$ being binary, we can formulate both constraints in a linear manner. Let $T^d \in \{0,1\}^{m \times n}$ be a simple transformation matrix, that rearranges q_t in such a way, that the i-th entry in $T^d q_t$ is the link-destination queue of link i, and define T^o in the same way for the link-origin queues. Then we get for the first constraint

$$\begin{aligned} \dot{v}_t &\le 2 \cdot \mathbf{1}_m - T^d \dot{q}_t \\ &= 2 \cdot \mathbf{1}_m - T^d q_0 - T^d R^+ \left(\operatorname*{diag}_{i=1,\dots m} \left\{ \omega_0^{[i]} \right\} \dot{v}_0 + \dots + \operatorname*{diag}_{i=1,\dots m} \left\{ \omega_{t-1}^{[i]} \right\} \dot{v}_{t-1} \right) \end{aligned} \tag{5.11}$$

and for the second

$$\begin{aligned} u_t &\le T^o q_t \\ &= T^o q_0 + T^o R^+ \left(\operatorname*{diag}_{i=1,\dots m} \left\{ \omega_0^{[i]} \right\} \dot{v}_0 + \dots + \operatorname*{diag}_{i=1,\dots m} \left\{ \omega_{t-1}^{[i]} \right\} \dot{v}_{t-1} \right) \end{aligned} \tag{5.12}$$

Defining a block triangular matrix as

$$\Delta \left(T^d R^+ \right) = \begin{pmatrix} 0 & & & \\ T^d R^+ & \ddots & & \\ \vdots & \ddots & \ddots & \\ T^d R^+ & \dots & T^d R^+ & 0 \end{pmatrix} \tag{5.13}$$

we can write this over the prediction horizon to yield

$$\left[I_{Hm} + \Delta \left(T^d R^+ \right) \Omega^E \right] \tilde{\dot{v}}_0 \le \mathbf{1}_H \otimes \left[2 \cdot \mathbf{1}_m - T^d q_0 \right] \tag{5.14}$$

for the first and

$$\left[I_{Hm} - \Delta\left(T^o R^+\right)\Omega^E\right]\tilde{\dot{v}}_0 \leq \mathbf{1}_H \otimes \left[T^o q_0\right] \tag{5.15}$$

for the second constraint. Together with the reliability constraint $\Omega^C \tilde{\dot{v}}_0 \geq \mathbf{1}$ and a suitable constituency constraint $\tilde{C}\tilde{\dot{v}}_0 \leq \mathbf{1}$ this completes evolution and constraints of the prediction-model. (In the end of this section, we will discuss the case, in which the constraints can not be fulfilled.)

This leaves us with the definition of a suitable objective function J. In a linear fashion, we use the weight vector $\gamma \in \mathbb{R}^n, \gamma < 0$ to assign rewards to filling any queue. The "closer" such a queue is to the subsystem-destination queue, the higher the reward it grants. With proper γ, the algorithm thus will automatically push the packet in the right direction. For simple networks, γ can be constructed by hand. How to arrive at an optimal γ is however still subject to research. In any case, the objective function becomes

$$J = \sum_{t=1}^{H} \gamma^\intercal \left(\dot{q}_t - q_0\right) = \left[\mathbf{1}_H^\intercal \otimes \gamma\right] \begin{pmatrix} R^+ & & \\ \vdots & \ddots & \\ R^+ & \dots & R^+ \end{pmatrix} \Omega^E \tilde{\dot{v}}_0 \tag{5.16}$$

To summarize, the control policy consists of solving the following minimization problem in *each* time step, while only applying the first component of the optimal control trajectory $\tilde{\dot{v}}_0$ (that minimizes the objective):

$$\begin{gathered} \min_{\tilde{u}} \ (5.16) \\ \text{s.t.} \\ \tilde{C}\tilde{\dot{v}}_0 \leq \mathbf{1} \\ -\Omega^C \tilde{\dot{v}}_0 \leq -\mathbf{1} \\ \left[I_{Hm} - \Delta\left(T^o R^+\right)\Omega^E\right]\tilde{\dot{v}}_0 \leq \mathbf{1}_H \otimes \left[T^o q_0\right] \\ \left[I_{Hm} + \Delta\left(T^d R^+\right)\Omega^E\right]\tilde{\dot{v}}_0 \leq \mathbf{1}_H \otimes \left[2\cdot\mathbf{1}_m - T^d q_0\right] \end{gathered} \tag{5.17}$$

Note that this is a binary linear program with linear constraints. Furthermore, any matrices can be pre-computed offline, making it feasible to solve. Given an optimal $\tilde{u}$ it is an easy task, to derive at the prediction of when packets will arrive at their corresponding destination.

For completeness we finally address some technicalities left open:

1. There are cases in which the reliability constraint can not be fulfilled by any $\tilde{\dot{v}}_0$ at all (e.g. if H is too small). As a solution, we append a dummy control $\dot{v}_D$ to $\tilde{\dot{v}}_0$, which has no influence on the prediction-model evolution and is penalized with suitable weights in J. Writing the reliability constraint as $\left[\tilde{\dot{v}}_0^\intercal \mid \dot{v}_D^\intercal\right]\left[\Omega^C C \mid I_m\right]^\intercal \geq \mathbf{1}^\intercal$ guarantees a feasible solution.

2. Once the subsystem-destination queue q^* has been filled, no further activations in this subsystem are to be scheduled. To this end, we engineer a dummy queue q^D and a link from q^* to q^D. We reward filling of q^D highly in J and disable the constraint so that it can be filled without limit. Making activation of the dummy link disjunct to any other activation in the subsystem will result in the desired behavior.

3. To ease the understanding we omitted a constraint that would force the policy to yield only reduced or equal delay times at consecutive time-steps. This constraint has to be added in order for the policy to stay consistent with its forecasts.

4. As a general way of defining γ, one can use the Dijkstra algorithm on each subsystem. Doing this, the weights of the links should be defined as the number of consecutive repetitions necessary to fulfill reliability over that link. Here, one can work with time-averaged transmission success probabilities. Offsetting the derived shortest paths for each queue then yields the reward coefficients for γ.

5. For the entire algorithm to work, we assume that all agents store their received packets until the network controller signals to alleviate them. Thus, in the system-model, we implicitly also work with R^+ instead of R.

5.2.4 Simulation

For numerical results, we compare the well known MW policy with our introduced Reliable Predictive Network Control policy (rPNC). We use a scenario in which three robots communicate via wireless connection as depicted in Figure 5.2 and all communication is routed through a central router. Disturbances in the communication are caused by periodic environmental effects, e.g. moving objects in a factory building.

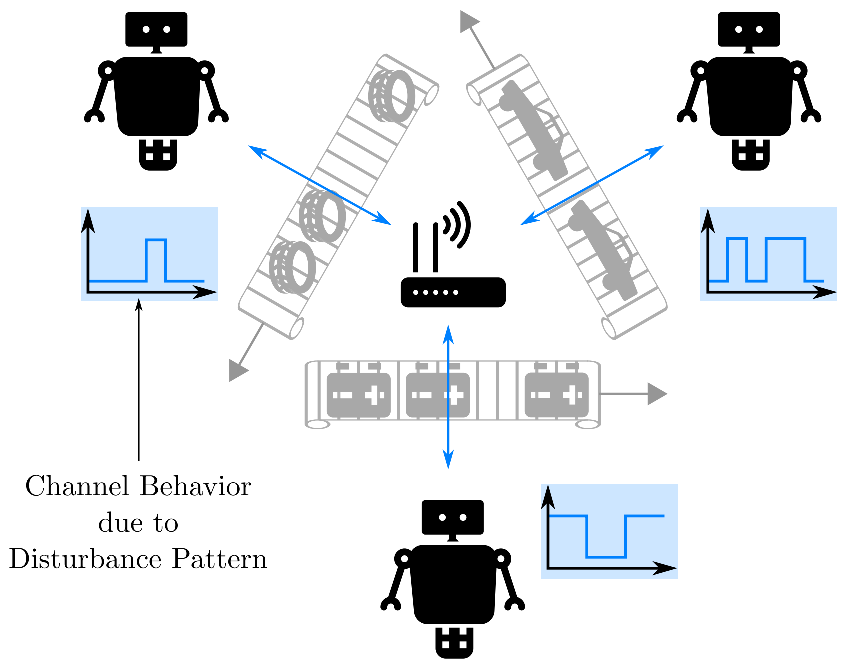


Fig. 5.2: Scenario used for simulations. The channel of each agent is subject to its individual disturbance. A prudent network controller has to leverage these in order to maximize throughput.

For their work tasks, the robots need to exchange data. We assume that at time $t = 0$, each robot needs to send its data (modeled as one packet) to the other two and has signaled this need to the router. The router then assigns communication resources to the robots. Specifically, we assume that in each time-step, *either* one, and only one robot may send its packet to the router (interference property) *or* the router may send a single packet to all robots at once (broadcast property). Note that we imply, that signaling between the agents is instantaneous compared to the transmission of the packets containing the actual data. This seems to be a reasonable assumption when working with RMPCs, since exchanged data consists of entire trajectories of their internal states.

The performance of rPNC depends highly on the periodic disturbances that dictate the transmission success probabilities $p_t^{[i]}$ (which are found as the entries of W^{s_t}). For this reason we simulate over many randomly selected disturbance patterns and then average the results. We assume that any disturbance pattern has a period of k steps and evolves deterministic in time so that it can be represented (in terms of transmission success probabilities) by a discrete-time Markov chain with binary transition matrix and a set $\mathcal{W}$ of probability matrices W^i, holding the p_t^i. Furthermore we specify, that in each step the transmission success probability is either high $\hat{p}$ or low $\check{p}$ resulting in $2^k - 1$ different patterns (we do not consider the unique pattern, only consisting of $\check{p}$). Finally, in order to avoid non-unique solutions to the optimization, we slightly vary $\hat{p}$ for each pattern once this pattern is selected for simulation, by using a randomly drawn value from a uniform distribution over the interval $[\hat{p} \pm 0.01\hat{p}]$ instead of the value $\hat{p}$ itself; the same goes for $\check{p}$.

A *single* simulation run follows the system evolution for N time steps while the rPNC policy uses a prediction horizon of H. We accumulate simulation runs via two loops. The first one repeats over x randomly chosen cases (without repetition). A case is defined as an assignment of patterns to the links of the three robots, resulting in $\left(2^k - 1\right)^3$ different cases. In a second loop (having a fixed case) we simulate over different initializations of the routing matrix (equivalent to different initializations of Bern$[W^{\sigma_t}]$ per link per time-step). We repeat this inner loop y times, resulting in $x \cdot y$ simulation runs. Parameter N is the number of time-steps over which a single simulation run is executed.

Detailed Description of a Specific Case

We first demonstrate the general disadvantage of MW on a specific case (i.e. every link has a fixed pattern assigned). We chose the following parameters:

$\hat{p}$	$\check{p}$	$1-\tau$	N	H	x	y
100%	0%	90%	20	4	1	1

Table 5.1: Simulation Parameters (Specific Case)

The patterns are illustrated in Figure 5.3. The blue colors indicate time-steps, in which $p_t^{[i]} = \hat{p} = 100\%$, gray colors indicate that $p_t^{[i]} = \check{p} = 0$. Robot 1 can only communicate once per period; Robot 2 twice. In the described setup, each robot has to send its packet over its link to the router (disjunct actions), before the router can possibly broadcast the packet, using two resource blocks at once. Hence, in the very first time-step, in order to minimize overall delay, it is always optimal to let Robot 1 send its packet to the router, since Robot 2 has the uncontested third time-step to do so and Robot 3 can only communicate in orthogonal time-steps. Using rPNC, this is indeed always the first action taken.

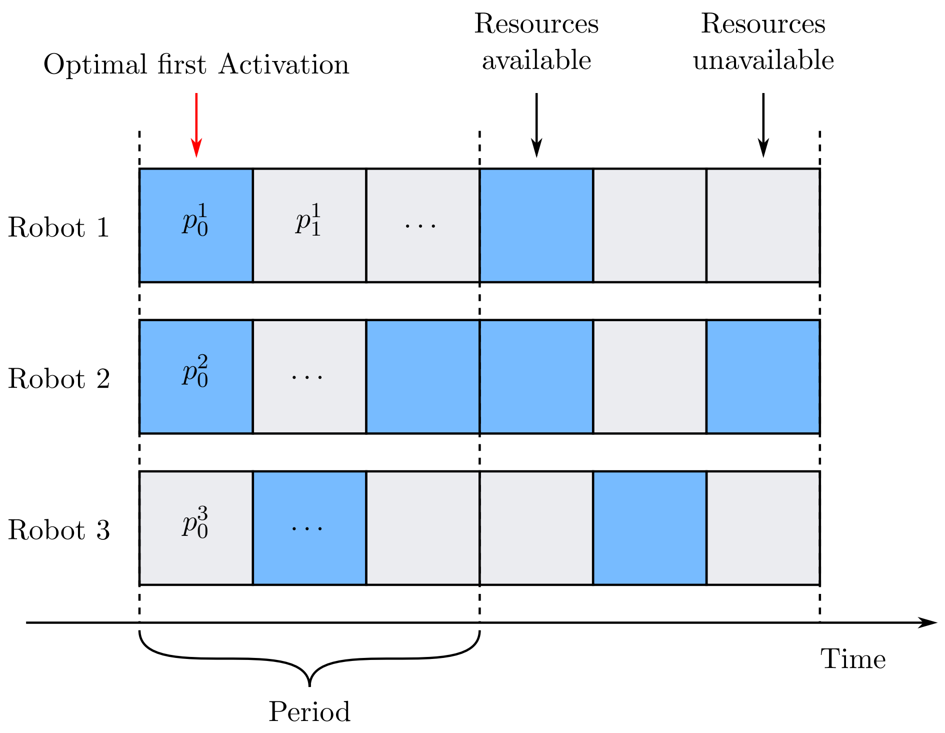


Fig. 5.3: Pattern assignment for a specific case/disturbance. Blue time-slots indicate that communication is possible, gray ones the opposite.

However, using MW, the decision which Robot (1 or 2) gets to communicate in the very first time-step only depends on the transmission success probabilities $p_0^{[1]}$ and $p_0^{[2]}$. In practice, if $p_0^{[2]}$ is just slightly higher than $p_0^{[1]}$, MW will allocate the first time-step to Robot 2, hence resulting in a sup-optimal control of the network.

To showcase this, we simulate over all possible slight variations in $\hat{p}$. The behavior of MW might differ, depending on how the robots $1, 2, 3$ must be mapped to the indices α, β, δ in order to fulfill the inequality $\hat{p}^\alpha > \hat{p}^\beta > \hat{p}^\delta$. There are six different mappings that do that, corresponding to the symmetry group S_3. Figure 5.4 shows simulation results for all six possibilities. MW and rPNC are compared by accumulating (over all robot-queues) their respective delays and taking the quotient. For this specific case, rPNC reduces overall delay by about 17 to 24 percent compared to MW, depending on the transmission success probabilities.

General Simulation Results

Next, we present results from extended simulations (Monte-Carlo simulation) over several cases.

$\hat{p}$	$\check{p}$	$1-\tau$	N	H	x	y
...	...	...	40	4	10	200

Table 5.2: Simulation Parameters (Monte-Carlo)

We simulate for different $\check{p}$, where we adjust $\hat{p}$ according to $\hat{p} + \check{p} = 1$; Figure 5.5 holds the results. Note that we also adjusted the threshold τ when using a different $\check{p}$, so that rPNC always deems $\hat{p}$ reliable, i.e. $\tau > 1 - \hat{p}$. Not adjusting τ leads to a distinct drop in performance of the rPNC policy, since the short horizon of $H = 4$

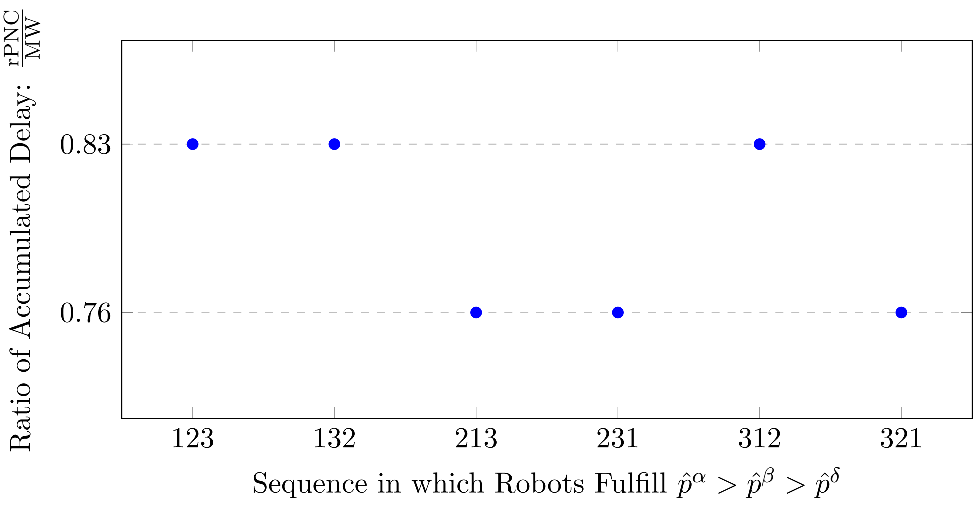


Fig. 5.4: Simulation results for a specific case.

does not suffice to schedule most of the transmission in a reliable way. In other words, a high reliability requirement (in comparison to the available transmission success probabilities), has to be accompanied with a far enough horizon to enable the algorithm to reliable schedule in its prediction-model.

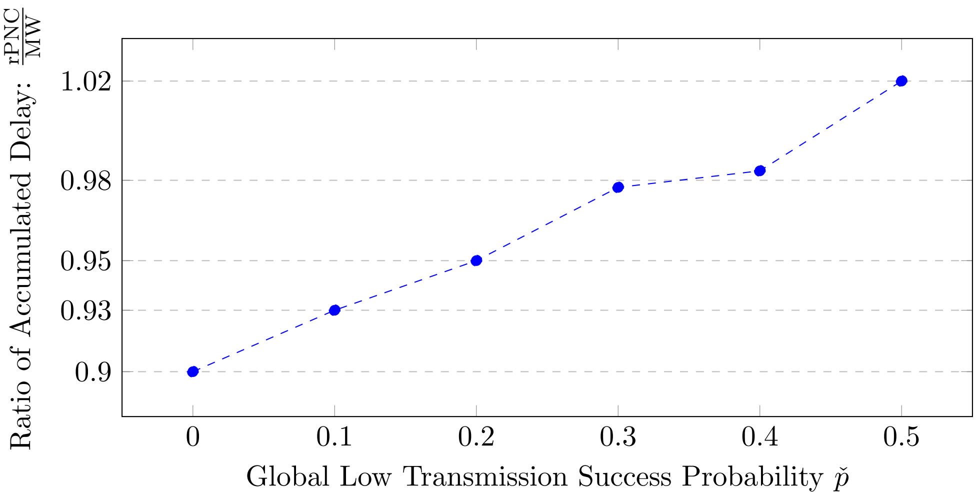


Fig. 5.5: Simulation results for the overall Monte-Carlo simulation.

The simulations show, that we can expect an average reduction in accumulated delays of about 10%, if transmission success probabilities (channel states) jump between 1 (superb) and 0 (not available at all). The closer $\check{p}$ and $\hat{p}$ get, the more this pleasant reduction diminishes. In the instance, that $\check{p} = \hat{p} = 0.5$, MW even exceeds the performance of rPNC. This result is due to the fact, that in this instance, future predictions are the least helpful (there is no time dependent pattern to take advantage of and the one step optimal control becomes the general optimal control).

Note that the resulting quotient of a single simulation can differ heavily (0.5 to 1.5) from the obtained averaged performance quotients (0.9 . . . 1.02). Also, the discussion above does not take into account, that we additionally yield individual forecasts of delays. One should keep in mind, that the reduction of accumulated delay is only one benefit of the rPNC algorithm. And finally, the simulated scenario resembles a bursty stimulation of the network. The disadvantages of MW become less stringent, once the bursty traffic transitions into a steady state traffic, because then, MW can use the length of individual queues to obtain information on good and bad paths through the network.

Time Consumption

By applying the binary linear optimization over the horizon H, the minimization problem in rPNC has to be solved for $m \cdot H$ unknown binary values. In comparison, MW does only solve for m unknown binary values, since in each step it solves

$$\min_{u} q^{\intercal} R v \tag{5.18}$$

where q and R are current queue vector and current routing matrix. Though one would intuitively suspect an exponential growth (with H) in time needed for deriving at an optimal solution, at least for scenario presented here, simulations suggest a linear growth as shown in Figure 5.6. The used parameters are captured in Table 5.3, where we chose $N = 7$ to ensure that there are always packets still to be transmitted. If all packets are transmitted, then the consecutive optimization is trivial which would in turn compromise the simulation results.

$\hat{p}$	$\check{p}$	$1-\tau$	N	H	x	y
70%	30%	68%	7	...	10	100

Table 5.3: Simulation Parameters (Time Consumption)

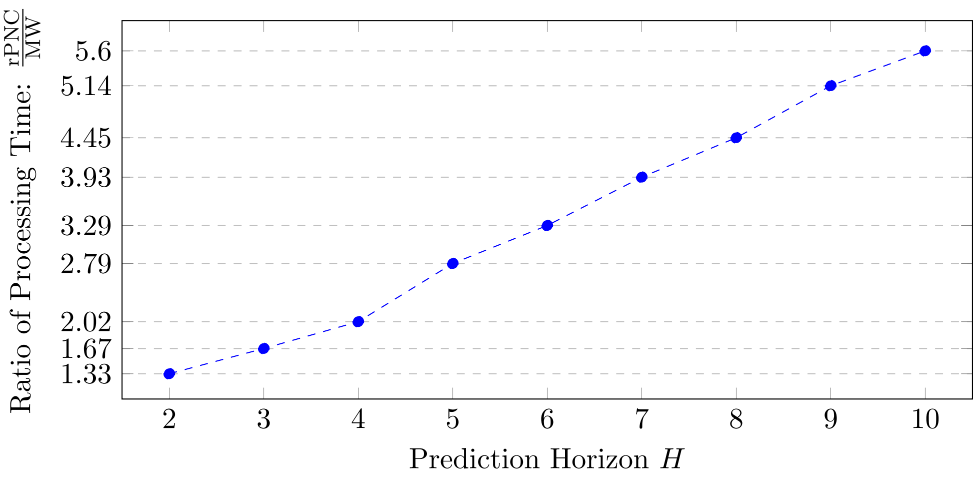


Fig. 5.6: Ratio of processing time as a function of the prediction horizon. Apparently, the processing time only grows linearly with the horizon, at last for small values.

Finally, we also try to investigate how time consumption scales with the number of subsystems in the prediction-model, i.e. with the number of packets to be transmitted simultaneously. We use again the parameter set from Table 5.3 but vary the number of packets, for which transfer is requested in the very first time-step; the results are shown in Figure 5.7. The casual decrease in time consumption with growing number of packets might be a consequence of the utilized optimizer (gurobi) applying a branch-and-bound procedure to solve the minimization. This remains to be analyzed. Nevertheless, the results once more suggest a linear growth in time consumption with increasing number of packets.

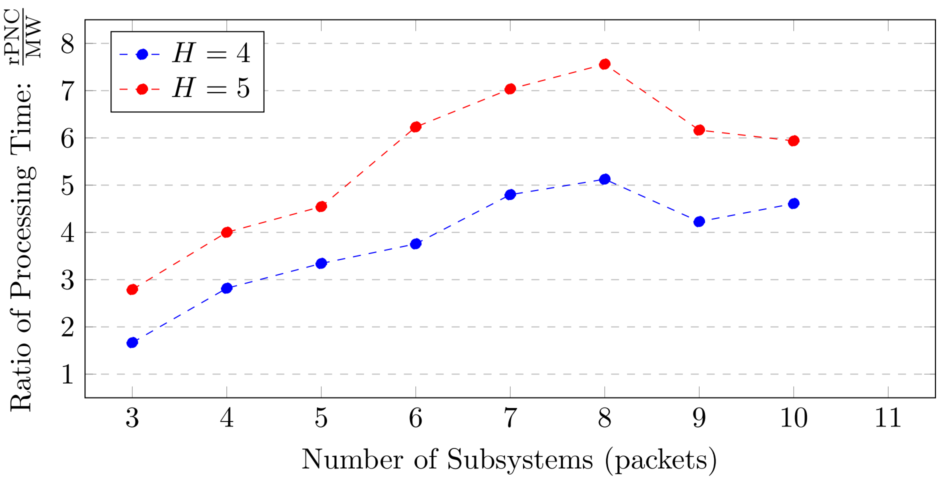


Fig. 5.7: Ratio of processing time as a function of the number of subsystems. Though the dependency does not seem to be linear, the processing time apparently does not scale exponentially with the number of subsystems (at least for small values).

5.2.5 Conclusion

We provided a proof of concept for a new network control policy, which is predictive in nature (based on MPC paradigms), and does provide reliable forecast of delay times of single data packets. The applied optimization problem is linear and thus quite feasible to implement. The numerical results show a clear advantage of our approach in comparison to MW when it comes to pure routing and scheduling decisions. However these advantages are leveraged with an increased utilization of computational resources, the dimension of which we could identify.

5.3 Concluding Remarks

Compared to Paper 4, the introduced algorithm in this paper allows for the repetition of links to take place in non-consecutive time-steps, which is a significant improvement. As before though, the paper misses an illustrative example in order to describe the intricate prediction-model. We will deliver such an example in this section and again use the queueing network from Figure 4.5, where 3 queues are connected via 2 links according to the following routing and transformation matrices

$$R = \begin{pmatrix} -1 & 0 \\ +1 & -1 \\ 0 & +1 \end{pmatrix}, \qquad T^o = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \qquad T^d = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.19}$$

This time, the transmission success probabilities are to be time-invariant with values $p_t^{[1]} = 0.8$ and $p_t^{[2]} = 0.9$. Choosing the failure-probability threshold to be $\tau = 0.1$, we end up with

$$\omega_t^{[1]} = 0.7, \qquad \omega_t^{[2]} = 1 \tag{5.20}$$

These ω-values are the continuous counterparts to the p-values from the queueing model. They dictate how much an activation will actually influence the prediction-model.

It is readily verified, that the prediction over 3 time-steps becomes

$$
\begin{aligned}
\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} &= \begin{pmatrix} q_0 \\ q_0 \\ q_0 \end{pmatrix} + \begin{pmatrix} R^+ & \mathbf{0} & \mathbf{0} \\ R^+ & R^+ & \mathbf{0} \\ R^+ & R^+ & R^+ \end{pmatrix} \begin{pmatrix} 0.7 & & & & & \\ & 1 & & & & \\ & & 0.7 & & & \\ & & & 1 & & \\ & & & & 0.7 & \\ & & & & & 1 \end{pmatrix} \tilde{\dot{v}}_0 \\
&= \begin{pmatrix} q_0 \\ q_0 \\ q_0 \end{pmatrix} + \left(\begin{array}{cc|cc|cc} 0 & 0 & & & & \\ +0.7 & 0 & & & & \\ 0 & +1 & & & & \\ \hline 0 & 0 & 0 & 0 & & \\ +0.7 & 0 & +0.7 & 0 & & \\ 0 & +1 & 0 & +1 & & \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ +0.7 & 0 & +0.7 & 0 & +0.7 & 0 \\ 0 & +1 & 0 & +1 & 0 & +1 \end{array} \right) \begin{pmatrix} \dot{v}_0^{[1]} \\ \dot{v}_0^{[2]} \\ \dot{v}_1^{[1]} \\ \dot{v}_1^{[2]} \\ \dot{v}_2^{[1]} \\ \dot{v}_2^{[2]} \end{pmatrix}
\end{aligned}
\tag{5.21}
$$

Note that any control-vector can only fill, but never deplete any queues, and that the first link only fills queues with 0.7 packets per activation.

According to (5.15), activating this first link is possible right out of the gate, since the first queue is initialized with a packet:

$$
\begin{pmatrix} \dot{v}_0^{[1]} \\ \dot{v}_0^{[2]} \end{pmatrix} \leq T^o q_0 = \begin{pmatrix} q_0^{[1]} \\ q_0^{[2]} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{5.22}
$$

If we extend this constraint to all time-steps of the prediction and consider the evolution, we get

$$
\left[\begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix} - \left(\begin{array}{cc|cc|cc} & & & & & \\ & & & & & \\ \hline 0 & 0 & & & & \\ 0.7 & 0 & & & & \\ \hline 0 & 0 & 0 & 0 & & \\ 0.7 & 0 & 0.7 & 0 & & \end{array} \right) \right] \begin{pmatrix} \dot{v}_0^{[1]} \\ \dot{v}_0^{[2]} \\ \dot{v}_1^{[1]} \\ \dot{v}_1^{[2]} \\ \dot{v}_2^{[1]} \\ \dot{v}_2^{[2]} \end{pmatrix} \leq \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{5.23}
$$

Indeed, the constraint is not too hard to translate. Let us take e.g. the last row:

$$
\dot{v}_2^{[2]} \leq 0 + 0.7\dot{v}_0^{[1]} + 0.7\dot{v}_1^{[1]} \tag{5.24}
$$

Activating $\dot{v}_t^{[2]}$ is only possible, if both $\dot{v}_0^{[1]}$ and $\dot{v}_1^{[1]}$ were active before since only then the RHS exceeds 1. This stems from the idea that with at least two activations, reliable transmission can be assumed because then the transmission failure probability is undercut. If the prediction horizon was larger, there would be more opportunities, e.g. $\dot{v}_2^{[1]}$, to make the RHS greater 1.

The constraint from above prevents links from activation if their origin queue has not been filled beyond 1 (i.e. did not receive enough transmissions to reliably contain the packet). However, by the virtue of having a linear objective, if there exists a motivation for the control to fill up a certain queue, it does not disappear (in contrast to e.g. quadratic objectives where a global extreme point exists). Hence, there also needs to be a countermeasure to restrain a link from being activated if its destination queue is already filled beyond 1 (bounding the domain of the optimization problem and thereby creating local extreme points). These constraints are given through (5.14) and do evaluate to

$$\left[\begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix} + \left(\begin{array}{cc|cc|cc} & & & & & \\ & & & & & \\ \hline 0.7 & 0 & & & & \\ 0 & 1 & & & & \\ \hline 0.7 & 0 & 0.7 & 0 & & \\ 0 & 1 & 0 & 1 & & \end{array}\right)\right] \begin{pmatrix} \dot{v}_0^{[1]} \\ \dot{v}_0^{[2]} \\ \dot{v}_1^{[1]} \\ \dot{v}_1^{[2]} \\ \dot{v}_2^{[1]} \\ \dot{v}_2^{[2]} \end{pmatrix} \leq \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix} \tag{5.25}$$

Exemplarily investigating the 5-th row yields

$$\dot{v}_2^{[1]} \leq 2 - 0.7\dot{v}_0^{[1]} - 0.7\dot{v}_1^{[1]} \tag{5.26}$$

which encodes that after activating the first link twice it can no longer be activated as the RHS is smaller than 1. And indeed 2 activations should be enough to have a packet reliably reach the queue $q^{[2]}$ (the destination of the link corresponding to $\dot{v}_t^{[1]}$).

Paper 6

Prediction of AoI

Full Paper Title Using AoI Forecasts in Communicating and Robust Distributed Model-Predictive Control

Publisher IEEE Transactions on Control of Network Systems (TCNS), 2021

Authors Jannik Hahn, Richard Schoeffauer, Gerhard Wunder, Olaf Stursberg

Acknowledgment This work is financially supported in part by the German Research Foundation (DFG) within the research priority program SPP1914 – *Cyber-Physical Networking.*

Abstract In order to enhance the performance of Cyber-Physical Systems, this paper proposes the integrated design of distributed controllers for distributed plants together with the control of the communication network. Conventional design methods use static interfaces between both entities and therefore rely on worst-case estimations of communication delay, often leading to conservative behavior of the overall system. By contrast, the present approach establishes a robust distributed Model-Predictive Control scheme, in which the local subsystem controllers operate under the assumption of a variable communication schedule that is predicted by a network controller. Using appropriate models for the communication network, the network controller applies a predictive network policy for scheduling the communication among the subsystem controllers across the network. Given the resulting time-varying predictions of the age of information, the paper shows under which conditions the subsystem controllers can robustly stabilize the distributed system. To illustrate the approach, the paper also reports on the application to a vehicle platooning scenario.

6.1 Preliminary Remarks

Paper 4 took a first attempt at making the predictions from the communication system available to the control realm. The intention behind this: enhancing the performance of the plant controllers by supplying them with these predictions. In the communication realm, Paper 4 uses a queueing framework since this was the initial set-up proposed in the project description. However, as the project advanced, we realized that the queueing framework is somewhat too complex for the practical scenarios we had in mind.

Specifically, it is quite reasonable to assume that distributed controllers, working together to achieve a common control goal, are connected in a very tight fashion, e.g. over a single base station, because there exists a clear motivation to invest in such communication solutions. If this is the case, however, communication over multiple nodes is of no concern. In the paper at hand, we therefore leave behind the queueing framework and concentrate on a most simple topology: one base station and various agents, connected to that base station via wireless communication channels. This enables us to change our focus: away from optimizing the routing decisions and towards optimizing the actual delay or rather Age-of-Information.

Please take note that since the paper at hand contains a large contribution from our research partners (Universität Kassel), we deviate from the notation guidelines that were established in the introduction of this dissertation. The details are clarified in the paper.

6.2 Paper Body

6.2.1 Introduction

Current technological advances in communication technology have lead to systems in which networks connect more and more locally controlled and autonomously operating devices. Such systems have an impact across a large number of applications, including networked automobile and traffic systems, smart energy grids, and the next generation of manufacturing plants (*industry 4.0*). Typically referred to as *Cyber-Physical Systems*, these systems are composed of physical components, digital and computational nodes, as well as the interconnecting communication infrastructure [62]. While traditional engineering concepts follow *divide-and-conquer* principles for separated and largely decoupled design of these components, requirements of high performance, reliability, as well as online and autonomous reconfiguration call for integrated design in which inter-dependencies are carefully taken into account [63].

This paper proposes a new approach of the latter type, tailored to the specific case of combining a wireless communication network with a distributed plant in which subsystems are locally controlled by model-predictive controllers (MPCs). The controllers aim at establishing cooperation with respect to a common cost functional formulated for the distributed plant, thus requiring to exchange data between the controllers across a centrally organized communication network with possibly time-varying properties of connectivity, reliability, and latency.

In order to let the controllers adapt to such properties autonomously and online, the use of model-predictive controllers is a straightforward choice: they do not only allow for the consideration of predicted behavior of other controlled entities and constraints for states, inputs, and outputs, but also imperfections in the communication. In fact, Model-Predictive Control without consideration of communication defects has reached a state of considerable maturity, including variants for nonlinear dynamics [64, 65], systems with uncertainties [66, 67, 68, 69], for fast computations [70, 71, 72], and distributed settings [73, 74, 75].

With respect to versions of distributed MPC taking network imperfections into account, solutions have been proposed in [76, 77, 54]. Common lines in these studies are, however, that network delays are either assumed to be negligibly small compared to the dominant plant dynamics [78, 79, 80], or that an upper bound of the delay (commonly named as *worst-case delay*) is assumed to be known [81, 82]. Own work in this direction has aimed at devising robust MPC strategies to compensate for the maximum delays [54], or to use schemes of event-based communication [50].

However, explicitly accounting for the network defects by use of a worst-case delay within robust control schemes is, most of the time, overly conservative

since delays are typically governed by a Chi-like distribution [83], such that the occurrence of the worst-case delay is extremely rare.

Consequently, the subsystem controllers (network agents) typically have much newer data available than what would be expected under the worst-case delay. In addition, a time-varying communication schedule leads to non-uniformly distributed instances of information reception and therefore lends itself to a description via the so-called *age of information* (AoI) metric that measures the time elapsed between generation and reception of information. This motivates to develop methods that can obtain and make use of the expected AoI, thus circumventing the static interface between communication and control that a worst-case delay typically amounts to. Assuming that a basic model for the link quality can be obtained (e.g. via machine learning techniques [84, 85]), this paper proposes a model-predictive network controller to handle both the management of transmissions as well as the prediction of future AoI, a strategy that is reminiscent of our prior work [25].

In contrast to that work, however, the paper at hand focuses on AoI rather than the throughput metric, shifting its focus to timely data arrival rather than transmission of large data streams. This also facilitates the utilization of a network topology in which routing problems disappear, since every agent is in proximity of any other agent. Such topologies are commonly encountered in related literature, where it stands to minimize the overall age of data in machine-to-machine communication scenarios [86, 87, 88].

Recent own work in [57] has sketched the idea of using predictive controllers for both the minimization of network delays, and the control of distributed plants under consideration of predicted AoI. That paper, however, neither detailed the network control scheme, nor the interface between network and subsystem controllers, nor the stability of the overall scheme. In contrast, the present paper proposes a novel predictive control scheme for the communication network and defines a mathematical interface through which the subsystem controllers can make use of the resulting forecasts of the AoI. Furthermore, it shows how these forecasts can be used to enhance control performance of the subsystem controllers, and under which conditions robust stability is ensured.

The paper is organized as follows: Section 6.2.2 introduces the general system architecture. Section 6.2.3 presents the design of a predictive network control scheme that generates delay forecasts. Section 6.2.4 presents the design of a distributed predictive control scheme making use of the delay forecasts, and stability of the scheme is proven. Section 6.2.5 illustrates the performance gain, when employing the proposed methods to a vehicle platoon scenario.

6.2.2 Set-Up and Notation

Set-up This paper considers Cyber-Physical Systems composed of two main parts, a distributed control system (CSYS) comprising a set of locally controlled subsystems, and a communication network (CNET) over which the local controllers of the subsystems in CSYS can communicate, see Figure 6.1. The dynamics of the subsystems are assumed to be decoupled in this setting, i.e. the state of one subsystem may not directly affect the dynamics of another, while different subsystems can impose constraints onto each other, and the behavior of one controlled subsystem may affect the control goal of another. An example in which such dependencies are practically relevant is that of a platoon of autonomous vehicles, as elaborated on in Section 6.2.5. It is further assumed that the local controllers are not able to measure the state of an interacting subsystem, thus information on neighbors can solely be obtained by exchange of information through the CNET.

In the latter, each subsystem controller acts as an agent in the communication network that requests data from and provides data to all other agents. Here, "data" refers to state information of the subsystems. The agents are connected via links which exhibit individual, time-depending behavior with respect to transmission quality. A centralized network controller manages a schedule, determining when which agent is allowed to broadcast its data to all other agents. In contrast to conventional network control approaches, this paper models the network controller as an MPC, enabling the generation of forecasts of future data transmission. In other words, it becomes possible to inform an agent of when it will receive data from another agent. The exploitation of this additional information is the key aspect leading to increased control performance for the distributed plant. To enable this exploitation, the CNET architecture foresees two layers of communication, as indicated in Figure 6.1: on one layer the agents broadcast their data to one another (blue arrows); on the other layer, the network controller informs the agents of both, the broadcast schedule and the forecasts of when information will arrive (red arrows).

Notation CSYS and CNET both operate on discrete-time domains with underlying time-steps. Any value of a discrete-time signal z at time $t_0 + k \cdot \Delta t$ is denoted by z_k with index $k \in \mathbb{N}$ and a constant interval $\Delta t \in \mathbb{R}^+$. A value z_{k+l} predicted in time k is indicated by $z_{k+l|k}$, where $z_{k|k} = z_k$. A complete predicted trajectory over a horizon of length H is denoted by $\tilde{z}_k = [z_{k|k}^\intercal,\ z_{k+1|k}^\intercal,\ \ldots,\ z_{k+H|k}^\intercal]^\intercal$. With slight abuse of notation, in some cases the first or last entry in the trajectory is omitted.

The symbol $\mathbf{z}_k$ refers to a stacked column vector of signals from different subsystems, and a trajectory of a stacked vector $\mathbf{z}_k$ is denoted by $\tilde{\mathbf{z}}_k$.

To refer to polytopic constraints for any element of $\tilde{\mathbf{z}}_k$: $\mathcal{Z} = \left\{\tilde{\mathbf{z}}_k \in \mathbb{R}^{\tilde{\mathbf{n}}_z} \middle| \tilde{\mathbf{C}}_z \cdot \tilde{\mathbf{z}}_k \leq \tilde{\mathbf{b}}_z\right\}$ the pair $(\tilde{\mathbf{C}}_z, \tilde{\mathbf{b}}_z)$ is used.

Matrices $0_{n,m}$ and $1_{n,m}$ denote $n \times m$-matrices of zeros and ones, respectively, while a column vector is simpler written as 0_n. For brevity, $\mathbf{0}$ is sometimes used to denote a zero matrix, if the dimensions are clear from the context.

Let an index set $\mathcal{N} = \{1, 2, \ldots, n\} \subset \mathbb{N}$ refer to a set of n subsystems. Then,

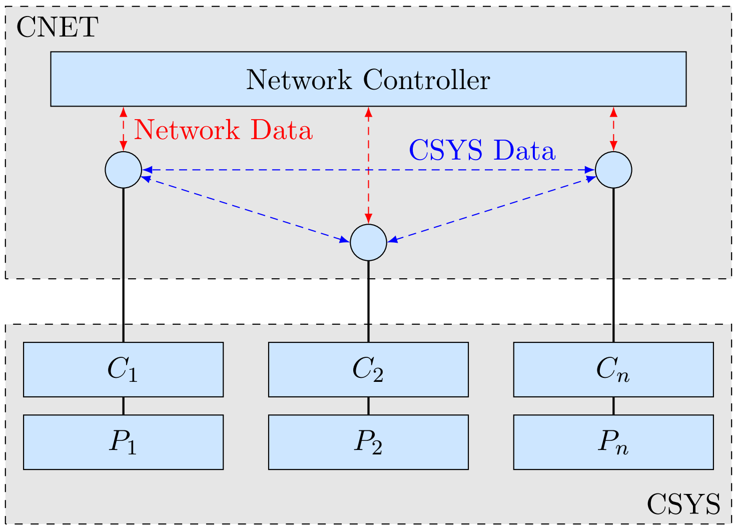


Fig. 6.1: Structure of the Cyber-Physical System, with local subsystem controllers C_i and plant subsystems P_i, and two different communication layers (red/blue).

a column vector z^i, a matrix A^i, and a set $\mathcal{A}^i$ indicate variables defined for the subsystem with index $i \in \mathcal{N}$. Furthermore, for the example of $\mathcal{N} = \{1, 2, 3\}$, the notation $\text{diag}\big([A^j], j \in \mathcal{N}\big)$ is equivalent to $\text{diag}(A^1, A^2, A^3)$, and $\text{prod}(\mathcal{A}^j, j \in \mathcal{N})$ defines the Cartesian product $\mathcal{A}^1 \times \mathcal{A}^2 \times \mathcal{A}^3$. In addition, $[z^j]_{j\in\mathcal{N}}$ defines the stacked vector $[z^{1^\intercal},\ z^{2^\intercal},\ z^{3^\intercal}]^\intercal$.

6.2.3 Communication Network

The CNET is modeled as a discrete-time packet-based system with n agents (corresponding to n subsystem controllers) and erroneous (wireless) transmissions. It is assumed that in each time-step each agent requires data from all other agents, and that likewise in each time-step each agent provides a new batch of data that can potentially be broadcasted to all other agents. The network resources are assumed to be limited, and thus agents typically have to work with out-dated data until new information is received. The age of information is pivotal for the agents' performance.

Definition 1. *Suppose each batch of data receives a time-stamp when being generated by its agent. The age of information $a_t^{ij} \in \mathbb{N}$ is the difference between the current time-step t and the time stamp of the latest batch of data that agent i received from agent j.*

If, in the current time-step t, agent i successfully receives data from agent j, a_t^{ij} is reset to 1 (since in time-discrete models, transmission and computation is assumed to take up an entire time-step). Otherwise a_t^{ij} is increased by 1. However, successfully receiving data from agent j can only occur, if (i) the network controller does schedule agent j to broadcast its data, via setting the control variable $v_t^j \in \{0, 1\}$ to 1, and if (ii) the data does not get lost due to erroneous transmissions (see Figure 6.2). The second part is expressed by the stochastic variable $p_t^{ji} \in \{0, 1\}$ (0 corresponding to failure) such that data is successfully received (by agent i from agent j) if $v_t^j \cdot p_t^{ji} = 1$. This results in the following evolution for the AoI:

$$a_{t+1}^{ij} = 1 + a_t^{ij}\big(1 - v_t^j p_t^{ji}\big) = 1 + a_t^{ij}\big(1 - p_t^{ji}\big)^{v_t^j} \qquad (6.1)$$

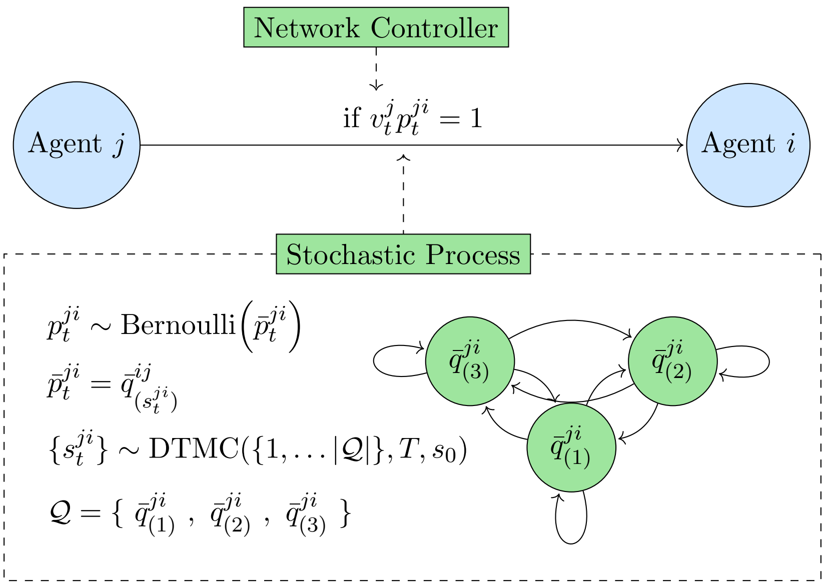

Network Controller
if $v_t^j p_t^{ji} = 1$
Agent j
Agent i
Stochastic Process
$p_t^{ji} \sim \text{Bernoulli}\left(\bar{p}_t^{ji}\right)$
$\bar{p}_t^{ji} = \bar{q}^{ij}_{(s_t^{ji})}$
$\{s_t^{ji}\} \sim \text{DTMC}(\{1, \dots |\mathcal{Q}|\}, T, s_0)$
$\mathcal{Q} = \{ \bar{q}^{ji}_{(1)} , \bar{q}^{ji}_{(2)} , \bar{q}^{ji}_{(3)} \}$
$\bar{q}^{ji}_{(3)}$
$\bar{q}^{ji}_{(2)}$
$\bar{q}^{ji}_{(1)}$

Fig. 6.2: Conditions for successful transmission from agent j to agent i: $v_t^j p_t^{ji}$ has to be 1; the DTMC stands exemplarily.

The control variables v_t^j are collected in the binary control-vector $v_t = [v_t^1, \dots v_t^n]^\intercal \in \{0,1\}^n$. Usually, e.g. due to interference properties, only certain agents are allowed to be engaged in broadcasting at the same time, and hence only certain realizations of v_t are admissible. The admissible set of control-vectors will be denoted by $\mathcal{V}$ such that $v_t \in \mathcal{V}$.

For ease of notation, the upper indices of the random variable p_t^{ji} are omitted for the moment: the process $\{p_t\}$ is governed by a Bernoulli process and a discrete-time Markov chain (DTMC) as shown in the bottom of Figure 6.2. The first one allows for consideration of unpredictable short-term drops in channel quality. In particular, each p_t is Bernoulli distributed (i.e. either 0 or 1) with success-parameter $\bar{p}_t \in [0,1]$ (note that the parameter is indeed time-variant). Opposed to that, the DTMC allows for consideration of a partially predictable, long-term behavior by dictating the time behavior of the parameter $\bar{p}_t$. To that end, let $\mathcal{Q} = \left\{\bar{q}_{(1)}, \dots \bar{q}_{(m)}\right\}$ be a finite set of values that $\bar{p}_t$ can take (the indices of the elements are set in parentheses to distinguish them from the time-step). Moreover, define $\{s_t\} \sim \text{DTMC}(\{1, \dots |\mathcal{Q}|\}, T, s_0)$ with transition matrix T and initial state s_0 such that in each time-step t it holds that $\bar{p}_t = \bar{q}_{(s_t)}$. For completeness, let the row vector σ_t denote the probability distribution to s_t. The following common assumption is made:

Assumption 2. *The quantities $\mathcal{Q}$ and T of the DTMC $\{s_t\}$ are known (e.g. as a result of machine learning techniques), and the state s_t is observable in time-step t (which is a common assumption for DTMCs).*

Note that under this model, it is possible to calculate the following expectation (needed later in (6.3)) for some arbitrary time-steps $t_0 < t_1 < \dots t_l$ in increasing order:

$$\mathbb{E}\left[\prod_{i=1}^{l} p_{t_i} \,\middle|\, \sigma_{t_0}\right] = \mathbb{E}\left[\prod_{i=1}^{l} \bar{p}_{t_i} \,\middle|\, \sigma_{t_0}\right] = \sigma_{t_0} \prod_{i=1}^{l} \left(T^{t_i - t_{i-1}} \Delta_{\mathcal{Q}}\right) \mathbf{1} \tag{6.2}$$

where $\Delta_{\mathcal{Q}}$ is the diagonal matrix whose entries are the elements of $\mathcal{Q}$ in the given order, and $\mathbf{1}$ is a suitably dimensioned row vector of ones.

An obvious control objective consists of minimizing the AoI for the *immediate next* time-step (a common approach in current network control strategies [89]). Opposed to that, our strategy aims to minimize the AoI over a certain number of time-steps, the prediction horizon N. Without loss of generality, assume that the current time-step is 0. Then, using the recursive form (6.1), the explicit expression for the AoI in time-step t becomes

$$\begin{aligned} a_t^{ij} &= 1 + \left(1 - v_{t-1}^j p_{t-1}^{ji}\right) \\ &\quad + \left(1 - v_{t-1}^j p_{t-1}^{ji}\right) \cdot \left(1 - v_{t-2}^j p_{t-2}^{ji}\right) + \dots \\ &\quad + \left(1 - v_{t-1}^j p_{t-1}^{ji}\right) \cdot \ \dots \ \cdot \left(1 - v_0^j p_0^{ji}\right) a_0^{ij} \\ &= t + \sum_{\mathcal{I} \in P(\mathbb{N}_{t-1})} (-1)^{|\mathcal{I}|} (a_0 + \min\{\mathcal{I}\}) \prod_{l \in \mathcal{I}} v_l^j p_l^{ji} \end{aligned} \tag{6.3}$$

Here, $P(\mathbb{N}_{t-1})$ is the power set of the set $\mathbb{N}_{t-1}$ (the natural numbers from 0 up to and including $t-1$) which stems from the products of $(1 - v_l p_l)$. It is $\min \emptyset = \max \emptyset = 0$ and $\prod_\emptyset = 1$.

Summing up (6.3) for $t = 1, \dots N$ and taking the expectation yields the suitable objective J^{ij} (the accumulated expected AoI values over the next N time-steps):

$$\begin{aligned} J^{ij}(\dot{v}_0, \dots \dot{v}_{N-1}) &:= \mathbb{E}\left[\sum_{t=1}^{N} a_t^{ij} \,\middle|\, a_0^{ij}, \sigma_0 \right] \\ &= \frac{N^2 + N}{2} + \sum_{\mathcal{I} \in P(\mathbb{N}_{N-1})} (N - \max\{\mathcal{I}\}) \\ &\quad \cdot (-1)^{|\mathcal{I}|} \left(a_0^{ij} + \min\{\mathcal{I}\}\right) \left(\prod_{l \in \mathcal{I}} \dot{v}_l^j \right) \mathbb{E}\left[\prod_{l \in \mathcal{I}} p_l^{ji} \,\middle|\, \sigma_0 \right] \end{aligned} \tag{6.4}$$

Notice how we employ $\dot{v}_t$ instead of v_t, which is necessary in order to distinguish the free variables over which the optimization is performed from the actual control variables employed to steer the system. Also, there is no need to consider the sum of squares of a_t^{ij} (which would be the usual objective) because every AoI is potentially reset to 1, no matter its value. Hence, larger values of the AoI will automatically be more prone to minimization than smaller ones, even in this linear formulation.

Still, (6.4) merely considers the AoI for data of agent i from agent j. The actual control objective however has to consider all AoI values in the entire system, and hence becomes:

$$\min_{\dot{v}_0, \dots \dot{v}_{N-1} \in \mathcal{V}} \sum_{i=1}^{n} \sum_{j=1}^{n} w^{ij} J^{ij}(\dot{v}_0, \dots \dot{v}_{N-1}) \tag{6.5}$$

where $w^{ij} \in \mathbb{R}_+$ are weights to balance for more or less important data. E.g., if agent i has no use for data from agent j then $w^{ij} = 0$. Eventually, these weights could also be signaled from the agents to the network controller, allowing for time-variant weights.

Due to the control-vector $\dot{v}_t$ being binary, this is a combinatorial problem with non-linear objective function. The amount of feasible solutions is given by $|\mathcal{V}|^N$ where $|\mathcal{V}|$ denotes the cardinality of $\mathcal{V}$. Furthermore, the amount of summands (over the power set in (6.4)) that need be evaluated in order to obtain the value of

J^{ij} for a single realization of $\dot{v}_0, \dots \dot{v}_{N-1}$ grows exponentially in N as well. While exhaustive enumeration leads to the optimal solution, one can expect the problem to quickly become intractable for higher dimensions in online solution. On the other hand, even for $N = 1$, satisfying analytical results on how to efficiently control the resulting Markov-decision-process do not yet exist [89]. Therefore, at this point, this papers proposes two heuristic relaxations to the optimization problem (6.5), that drastically reduces the effort of finding good approximations of the solution (as substantiated through extensive simulation).

First, the branching of the paths of the DTMC is replaced by a mean distribution. This means, in particular, that $\mathbb{E}\big[p_{t_1}^{ji} \cdot p_{t_2}^{ji} \,\big|\, \sigma_0\big]$ is approximated by $\mathbb{E}\big[p_{t_1}^{ji} \,\big|\, \sigma_0\big] \cdot \mathbb{E}\big[p_{t_2}^{ji} \,\big|\, \sigma_0\big]$. Though this procedure makes use only of a small share of the information encoded by the DTMC in principle, the scheme provides more and better information than a setting without prediction. Good approximations are especially obtained for cases in which the DTMC models slower-paced processes (i.e. in which the transition matrix can be transformed such that entries close to the diagonal are large). This fits well to the predictive controllers proposed for the CSYS, since such controllers are typically used for exactly such processes.

Using the substitution

$$\phi_t^{ji}(v_t) := \mathbb{E}\big[1 - v_t^j p_t^{ji} \,\big|\, \sigma_0\big] \in \{1, 1 - \sigma_0 T^t \Delta_Q \mathbf{1}\} \tag{6.6}$$

(where t in T^t is indeed an exponent) makes it possible to state (6.4) in a simplified form (directly derived from (6.3)):

$$\begin{aligned} I^{ij}(\dot{v}_0, \dots \dot{v}_{N-1}) &:= \mathbb{E}\left[\sum_{t=1}^{N} a_t^{ij} \,\middle|\, a_0^{ij}, \sigma_0\right]_{\text{relaxed}} \\ &= N + \sum_{t=1}^{N}\left(\sum_{l=1}^{t-1}\prod_{m=1}^{l}\phi_{t-m}^{ji}(\dot{v}_{t-m}) + a_0^{ij}\prod_{m=0}^{t-1}\phi_m^{ji}(\dot{v}_m)\right) \end{aligned} \tag{6.7}$$

Compared to the strict formulation, only $\frac{N^2}{2}$ terms need to be evaluated in order to obtain I^{ij} for a single realization of $\dot{v}_0, \dots \dot{v}_{N-1}$.

As a second relaxation, problem (6.5) is separated into N consecutive minimization problems:

$$\min_{\substack{\dot{v}_{t_N} \in \mathcal{V} \\ t_N \in \mathbb{N}_{N-1} \setminus \{t_1, \dots t_{N-1}\}}} \dots \min_{\substack{\dot{v}_{t_2} \in \mathcal{V} \\ t_2 \in \mathbb{N}_{N-1} \setminus \{t_1\}}} \min_{\substack{\dot{v}_{t_1} \in \mathcal{V} \\ t_1 \in \mathbb{N}_{N-1}}} \sum_{i=1}^{n}\sum_{j=1}^{n} w^{ij} I^{ij}(\dot{v}_0, \dots \dot{v}_{N-1}) \tag{6.8}$$

In each minimization, the most promising realization of v_t in the most promising time-step t is fixed and applied to the objective. However, in any subsequent minimization, this time-step then becomes unavailable and thus the relative feasible set for each subsequent minimization shrinks. This scheme results in $(N + (N-1) + \ldots + 1) \cdot |\mathcal{V}| = \frac{N^2+N}{2} \cdot |\mathcal{V}|$ evaluations of the objective before finding the solution. Both relaxations therefore drastically reduce the complexity of the problem and allow for a fast calculation of a sub-optimal solution. Note that choosing $N = 1$ still leads to the *exact* minimization over all *immediate* AoI values in the entire system.

In summary, the network controller operates as an MPC, solving (6.8) in every time-step t, with (6.8) being denoted relative to the said time-step. Let us revoke this relative notation and assume the current time-step to be t again. Considering more than just the AoI of the immediate next time-step in the control objective, naturally improves the control performance. However, a solution of (6.8) now also enables the network controller to predict when new data is probably arriving at the agents. Hereafter, these predictions are called "forecasts" and they are defined as trajectories of the predicted AoI:

$$\tilde{\dot{a}}_1^{ij} = [\dot{a}_1^{ij}, \dot{a}_2^{ij}, \dots \dot{a}_N^{ij}]^\intercal \tag{6.9}$$

Given a current network control trajectory $\tilde{\dot{v}}_0 = [\dot{v}_0^\intercal, \dots \dot{v}_{N-1}^\intercal]^\intercal$, forecasts are generated by the network controller based on the accumulated transmission failure probability by time-step $t+h$, which is:

$$\prod_{l=0}^{h-1} \mathbb{E}\left[\left(1 - p_{t+l}^{ji}\right)^{\dot{v}_{t+l}^{j}} \middle| \sigma_t \right] \tag{6.10}$$

As soon as this probability falls beneath a certain threshold τ, the network controller assumes that transmission did succeed at least once by the end of that time-step. Let t_τ^{ij} be the time-step in which this happens and let t_f^{ij} be the first time-step in which transmission was attempted:

$$t_\tau^{ij} := \min_h \left\{ h : \prod_{l=0}^{h-1} \mathbb{E}\left[\left(1 - p_{t+l}^{ji}\right)^{\dot{v}_{t+l}^{j}} \middle| \sigma_t \right] < \tau \right\} \tag{6.11}$$

$$t_f^{ij} := \min_h \left\{ h : \prod_{l=0}^{h-1} \mathbb{E}\left[\left(1 - p_{t+l}^{ji}\right)^{\dot{v}_{t+l}^{j}} \middle| \sigma_t \right] < 1 \right\} \tag{6.12}$$

Then, as depicted in Figure 6.3, the forecast $\tilde{\dot{a}}_t^{ij}$ becomes:

$$\tilde{\dot{a}}_t^{ij} = \begin{bmatrix} \dot{a}_{t+1}^{ij} \\ \vdots \\ \dot{a}_{t+t_\tau^{ij}-1}^{ij} \\ \dot{a}_{t+t_\tau^{ij}\pm 0}^{ij} \\ \dot{a}_{t+t_\tau^{ij}+1}^{ij} \\ \vdots \end{bmatrix} = \begin{bmatrix} \dot{a}_t^{ij} + 1 \\ \vdots \\ \dot{a}_t^{ij} + t_\tau^{ij} - 1 \\ \dot{a}_t^{ij} + t_\tau^{ij} \\ t_\tau^{ij} - t_f + 1 \\ \vdots \end{bmatrix} \tag{6.13}$$

The concept can be extended in a straightforward way for cases in which the threshold is transgressed more than once.

Once generated, those forecasts are transmitted to the agents. However, due to the underlying stochastics, they are only of limited reliability. Decreasing the threshold τ improves this reliability, but also reduces the amount of foreseen packet deliveries that can be detected within the fixed prediction horizon N. Hence, the right parametrization of τ depends on the agents needs.

Instead of investigating this parameterization, the paper at hand focuses on establishing a framework with which the network agents can make use of the AoI forecasts at all. Therefore, the agents are identified with a system of distributed, robust MPCs in a *deterministic* setting, explicitly adapted for the use of AoI forecasts. In order to overcome the apparent modeling mismatch between a *stochastic* communication but *deterministic* control framework, the following assumption is important:

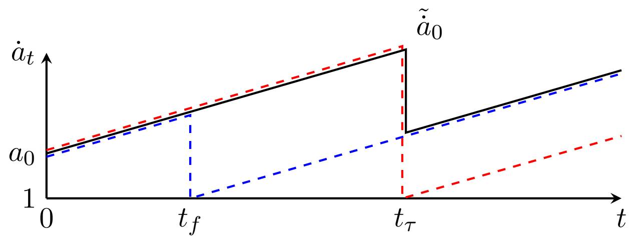


Fig. 6.3: Forecast generation when t_f and t_τ are given. The forecast (black line) results from combining the largest AoI from the two cases in which transmission succeeds in either in t_f (blue) or in t_τ (red).

Assumption 3. *The parameter τ is chosen sufficiently small, such that the forecasts are reliable, i.e. $\mathring{a}_t^{ij} \geq a_t^{ij}$ holds.*

Note that this assumption is similarly restrictive as the commonly used assumption of a worst-case delay, but additionally requires the network controller to not violate any previously communicated forecast.

6.2.4 Distributed Control System

With respect to the distributed plant to be controlled, consider again a set $\mathcal{N} = \{1, \ldots, n\}$ of discrete-time, linear time-invariant subsystems with the following dynamics:

$$x_{k+1}^i = A^i x_k^i + B^i u_k^i, \quad i \in \mathcal{N}. \tag{6.14}$$

For the subsystem with index i, x_k^i denotes the state vector and u_k^i the input vector, both subject to polytopic constraints:

$$x_k^i \in \mathcal{X}^i = \left\{ x \in \mathbb{R}^{n_x^i} \middle| C_x^i \cdot x \leq b_x^i \right\}, \tag{6.15}$$

$$u_k^i \in \mathcal{U}^i = \left\{ u \in \mathbb{R}^{n_u^i} \middle| C_u^i \cdot u \leq b_u^i \right\}. \tag{6.16}$$

It is assumed that the subsystems are coupled through constraints and the control objective, but not through the dynamics (6.14) at this stage.
Example: For the scenario of a platoon of vehicles, as will be considered in Section 6.2.5, coupling through constraints is understood as keeping the position of a subsequent vehicle strictly behind the position of the preceding vehicle (with a safety margin) in order to avoid collision. The coupling through the control objective can here be understood as minimizing the deviation from a desired distance between vehicles.

Let the index-set $\mathcal{N}^i \subset \mathcal{N}$ contain the indices of all subsystems coupled to i in this form, called *neighbors*. Since the exchange of information is limited to the communication network, each subsystem estimates the behavior of its neighbors based on the initially provided model of the neighbors' dynamics. During system operation, each subsystem controller receives or holds possibly outdated information on the neighbors' states, and estimates their expected current state by applying forward time-shift to the neighbors' models. Thus, a local, augmented (and now dynamically coupled) model can be constructed for any subsystem:

$$\mathbf{x}^i_{k+l+1|k} = \mathbf{A}^i \mathbf{x}^i_{k+l|k} + \mathbf{B}^i u^i_{k+l|k} + \mathbf{B}^i_1 \mathbf{u}^i_{k+l|k} + \mathbf{B}^i_2 \delta \mathbf{u}^i_{k+l|k}. \tag{6.17}$$

In here, the augmented state vector is $\mathbf{x}^i_k = \left[x^i_k; \left[x^{ij}_k\right]_{j\in\mathcal{N}^i}\right] \in \mathbb{R}^{\mathbf{n}^i_x}$, the augmented vector of nominal inputs of the neighbors is $\mathbf{u}^i_k = \left[u^{ij}_k\right]_{j\in\mathcal{N}^i}$, and possible deviations of the neighbors' inputs from their nominal values are denoted by $\delta\mathbf{u}^i_k = \left[\delta u^{ij}_k\right]_{j\in\mathcal{N}^i}$. Such a possible deviation of an input in time-step $k+l$ as predicted in time-step k is defined by:

$$u^j_{k+l} = u^{ij}_{k+l|k} + \delta u^{ij}_{k+l|k}. \tag{6.18}$$

The matrices of the model (6.17) follow as described in [57], e.g. if $\mathcal{N}^i = j$:

$$\mathbf{A}^i = \begin{bmatrix} A^i & 0 \\ 0 & A^j \end{bmatrix}, \mathbf{B}^i = \begin{bmatrix} B^i \\ 0 \end{bmatrix}, \mathbf{B}^i_1 = \mathbf{B}^i_2 = \begin{bmatrix} 0 \\ B^j \end{bmatrix}. \tag{6.19}$$

All subsystems $j \in \mathcal{N}^i$ communicate conservative uncertainty sets $\Delta\mathcal{U}$ for their possible deviations to subsystem i, i.e. their selected inputs are guaranteed to be contained in these sets. Collecting all the communicated data, polytopic constraints for subsystem i are written as:

$$\delta\mathbf{u}^i_{k+l|k} \in \delta\mathbb{U}^i_{k+l|k} = \text{prod}\left(\Delta\mathcal{U}^{ij}_{k+l|k}, j \in \mathcal{N}^i\right) \tag{6.20}$$

where the sets $\Delta\mathcal{U}^{ij}_{k+l|k}$ correspond to the communicated sets but shifted forward in time to the current time-step. Note, that also subsystem i has to communicate such an uncertainty set to all its neighbors. The determination of this uncertainty set is described later in (6.37).

As already mentioned, the augmented model (6.17) is dynamically coupled to neighbored subsystems such that coupling by state constraints can be formulated by:

$$\mathbf{x}^i_{k+l|k} \in \mathbb{X}^i = \left\{\mathbf{x} \middle| \mathbf{C}^i_x \cdot \mathbf{x} \le \mathbf{b}^i_x\right\}. \tag{6.21}$$

Extending the model (6.17) to the prediction-model (c.f. [57]) the behavior and constraints of the augmented system are predicted up to the control horizon H:

$$\tilde{\mathbf{x}}^i_k = \tilde{\mathbf{A}}^i \mathbf{x}^i_{k|k} + \tilde{\mathbf{B}}^i \tilde{u}^i_k + \tilde{\mathbf{B}}^i_1 \tilde{\mathbf{u}}^i_k + \tilde{\mathbf{B}}^i_2 \delta\tilde{\mathbf{u}}^i_k \tag{6.22}$$

with $\tilde{u}^i_k$ as the input trajectory of subsystem i. The state trajectory $\tilde{\mathbf{x}}^i_k$ needs to satisfy

$$\tilde{\mathbf{x}}^i_k \in \tilde{\mathbb{X}}^i = \mathbb{X}^i \times \ldots \times \mathbb{X}^i_\xi = \left\{\tilde{\mathbf{x}} \middle| \tilde{\mathbf{C}}^i_x \cdot \tilde{\mathbf{x}} \le \tilde{\mathbf{b}}^i_x\right\} \tag{6.23}$$

for all possible deviations of neighbored subsystems:

$$\delta\tilde{\mathbf{u}}^i_k \in \delta\tilde{\mathbb{U}}^i_k = \left\{\delta\tilde{\mathbf{u}} \middle| \tilde{\mathbf{C}}^i_\delta \cdot \delta\tilde{\mathbf{u}} \le \tilde{\mathbf{b}}^i_{\delta_k}\right\}, \tag{6.24}$$

where the set $\delta\tilde{\mathbb{U}}_k^i$ is derived from (6.20).

To compensate for these possible deviations from preceding neighbors, the control law of each subsystem is formulated as a disturbance feedback in stacked vector form:

$$\tilde{u}_k^i = \tilde{v}_k^i + \tilde{\Delta}_k^i = \tilde{v}_k^i + K_k^i \delta\tilde{\mathbf{u}}_k^i \tag{6.25}$$

with $\tilde{v}_k^i$ as nominal input trajectory and $\tilde{\Delta}_k^i$ as disturbance feedback term. Note that the $\tilde{\Delta}_k^i$ vanishes if all neighbored subsystems do not deviate from communicated trajectories, and that the local input trajectory needs to satisfy a local time-varying constraint:

$$\tilde{u}_k^i \in \tilde{\mathcal{U}}_k^i = \left\{ \tilde{u} \middle| \tilde{C}_u^i \cdot \tilde{u} \leq \tilde{b}_{u_k}^i \right\} \subseteq \mathcal{U}^i \times \ldots \times \mathcal{U}^i. \tag{6.26}$$

Even though (6.16) is time-invariant, the time-variance of (6.26) results from a self-updating mechanism to be formulated later in (6.38). Thus, the control trajectory to be computed in k is constrained by the information about possible deviations from the nominal trajectory computed and communicated to connected subsystems in a previous time-step.

Proposition 1. *In each time-step k, the network-controller provides by* (6.9) *and Asm. 3 reliable forecasts of the future age of information $\tilde{a}_k^{ij}$ of each neighbor $j \in \mathcal{N}^i$ such that:*

1. *The disturbance feedback matrix K_k^i satisfies the structure:*

$$K_k^i = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \eta^i_{k+1,k-\bar{a}_k^i+1|k} & \cdots & \cdots & \mathbf{0} \\ \eta^i_{k+2,k-\bar{a}_k^i+1|k} & \ddots & \cdots & \mathbf{0} \\ \vdots & \ddots & \cdots & \mathbf{0} \\ \eta^i_{k+H-1,k-\bar{a}_k^i+1|k} & \cdots & \eta^i_{k+H-1,k+H-2|k} & \mathbf{0} \end{bmatrix} \tag{6.27}$$

with

$$\bar{a}_k^i = \max_{j\in\mathcal{N}^i}\left(a_k^{ij}\right), \tag{6.28}$$

$$\eta^i_{n,m|k} = \left[\eta^{ij\intercal}_{n,m|k}\right]^{\intercal}_{j\in\mathcal{N}^i} \in \mathbb{R}^{n_u^i \times \mathbf{n}_u^i},$$

$$\eta^{ij}_{n,m|k} \in \begin{cases} \mathbb{R} & \textit{if } k - a_k^{ij} < m \leq n - a^{ij}_{n|k} \\ \mathbf{0} & \textit{else.} \end{cases} \tag{6.29}$$

2. *If $N < H$, then $a^{ij}_{k+l|k} := \bar{a}^{ij}\ \forall l \in \{N+1,\ H\}$, where $\bar{a}^{ij} = \max_k\left(a_k^{ij}\right)$, and N is the horizon for the trajectory $\tilde{a}_k^{ij}$ as provided by the network controller.*

By substituting (6.25) in (6.22), the constraints (6.23) and (6.26) are combined into the set of all admissible input trajectories (c.f. [57]):

$$\left(\tilde{v}_k^i, K_k^i\right) \in \Pi_k^i\left(\mathbf{x}_k^i\right). \tag{6.30}$$

From [54], the specification of local control goals with respect to the augmented state and input vectors is adopted, establishing a form of coupling through control objectives.

The local control objective with finite horizon H and stage cost $L^i(x,u) = \|x\|^2_{Q^i_x} + \|u\|^2_{Q^i_u}$ is given by:

$$J^i_k = V^i_f\left(\xi^i_{k+H|k}\right) + \sum_{l=0}^{H-1} L^i\left(\mathbf{x}^i_{k+l|k}, \left[v^{i\intercal}_{k+l|k}\ \mathbf{u}^{i\intercal}_{k+l|k}\right]^\intercal\right) \tag{6.31}$$

with $Q^i_x = Q^{i\intercal}_x > 0, Q^i_u = Q^{i\intercal}_u > 0$, end cost function $V^i_f(r) = \|r\|^2_{P^i_\xi}$ with terminal cost $P^i_\xi = P^{i\intercal}_\xi \geq 0$, and terminal state ξ^i_k:

$$\xi^i_{k+H|k} = \left[\mathbf{x}^{i\intercal}_{k+H-\bar{a}^i+1|k}, \dots, \mathbf{x}^{i\intercal}_{k+H-1|k}, \mathbf{x}^{i\intercal}_{k+H|k}\right]^\intercal \tag{6.32}$$

In here, $\bar{a}^i = \max_{k,j\in\mathcal{N}^i}\left(a^{ij}_k\right)$ is the maximally possible age of information.

As is common in constrained MPC, the existence of a terminal set $\mathbb{X}^i_\xi$ is assumed in addition. The special requirements occurring from the time-varying communication are stated as follows:

Assumption 4. *Given a terminal set $\mathbb{X}^i_\xi \subseteq \mathbb{X}^i \times \ldots \times \mathbb{X}^i$ for the terminal state ξ^i_k, it holds that:*

1. *The set $\mathbb{X}^i_\xi$ is robust forward invariant with respect to the closed-loop system:*

$$\xi^i_{k+1} = A^i_\xi \xi^i_k + B^i_\xi \mathbf{u}^i_k \tag{6.33}$$

with

$$A^i_\xi = \left(\begin{bmatrix}\mathbf{0} & I \\ \mathbf{0} & \mathbf{A}^i\end{bmatrix} + \begin{bmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{B}^i K^i & \mathbf{0}\end{bmatrix}\right),\ B^i_\xi = \begin{bmatrix}\mathbf{0} \\ \mathbf{B}^i_1\end{bmatrix} \tag{6.34}$$

such that $\xi^i_{k+1} \in \mathbb{X}^i_\xi$ if $\xi^i_k \in \mathbb{X}^i_\xi$ for all $\mathbf{u}^i_k \in \operatorname{prod}(\mathcal{U}^j, j \in \mathcal{N}^i)$.

2. *The terminal state feedback $u^i_k = K^i\mathbf{x}^i_{k-\bar{a}^i+1} = \left[K^i\ \mathbf{0}\right]\xi^i_k = K^i_\xi \xi^i_k$ satisfies the local input constraint $u^i_k \in \mathcal{U}^i$ for all $\xi^i_k \in \mathbb{X}^i_\xi$.*

3. *The terminal cost P^i_ξ solves the Lyapunov equation $A^{i\intercal}_\xi P^i_\xi A^i_\xi - P^i_\xi + Q^i_\xi \preccurlyeq 0$ with $Q^i_\xi = \operatorname{diag}\left(\begin{bmatrix}\mathbf{0} \\ K^i\end{bmatrix}^\intercal Q^i_u \begin{bmatrix}\mathbf{0} \\ K^i\end{bmatrix}, \mathbf{0}, \ldots, \mathbf{0}, Q^i_x\right)$ and with respect to the autonomous and undisturbed closed-loop system:*

$$\xi^i_{k+1} = A_\xi \xi^i_k. \tag{6.35}$$

4. *$L(\mathbf{x}^i_{k+l|k}, \mathbf{0}) = 0$ for all $l \in \{H - a^i + 1, \ldots, H\}$ implies $\left\|\xi^i_{k+H|k}\right\|^2_{P^i_\xi} = 0$.*

Minimizing the cost function (6.31) with respect to the set of admissible inputs (6.30), the optimization problem to be solved in every time-step is the following:

$$\begin{aligned} V^i_k = \min_{\left(\tilde{v}^i_k, K^i_k\right)} & \left\|\xi^i_{k+H|k}\right\|^2_{P^i_\xi} + \sum_{l=0}^{H-1} \left\|\mathbf{x}^i_{k+l|k}\right\|^2_{Q^i_x} + \left\|\left[v^{i\intercal}_{k+l|k}\ \mathbf{u}^{i\intercal}_{k+l|k}\right]^\intercal\right\|^2_{Q^i_u} \\ \text{s.t.:}\quad & \mathbf{x}^i_{k+l+1|k} = \mathbf{A}^i\mathbf{x}^i_{k+l|k} + \mathbf{B}^i v^i_{k+l|k} + \mathbf{B}^i_1 \mathbf{u}^i_{k+l|k}, \\ & \text{and (6.30).} \end{aligned} \tag{6.36}$$

Solving (6.36) yields the nominal input trajectory $\tilde{v}_k^i$ and the feedback matrix K_k^i. Together with the bounds on $\delta\tilde{\mathbf{u}}_k^i$, the feedback matrix provides an upper bound for $\tilde{\Delta}_k^i$, such that:

$$\Delta\tilde{\mathcal{U}}_k^i = \left\{ \tilde{\Delta}_k^i \middle| \tilde{C}_u^i \tilde{\Delta}_k^i \leq \tilde{\gamma}_k^i \right\} \tag{6.37}$$

is the uncertainty set of the nominal input trajectory of subsystem i (for exact computation of $\tilde{\gamma}_k^i$ c.f. [54]).

The current state, nominal input trajectory, and uncertainty set determine the behavior of subsystem i for the next H time-steps, and are communicated to all neighboring subsystems. Forcing subsystem i in the next time-step $k+1$ to comply with the communicated information in time-step k, the time-varying input trajectory constraint (6.26) needs to be shifted forward in time by one time-step:

$$\tilde{b}_{u_{k+1}}^i = \tilde{C}_u^i \left[v_{k+1|k}^{i\intercal}, \ldots, v_{k+H-1|k}^{i\intercal}, \mathbf{0} \right] + \left[\gamma_{k+1|k}^{i\intercal}, \gamma_{k+H-1|k}^{i\intercal}, \ldots, b_u^{i\intercal} \right]. \tag{6.38}$$

Theorem 4. *If Asm.s 3 and 4 hold, and* (6.36) *is feasible for all subsystems $i \in \mathcal{N}$ in time-step $k = 0$,* (6.36) *remains feasible for all subsystems and $k > 0$.*

Proof. If (6.36) is feasible in time-step k, it provides through (6.30) a nominal input $v_{k|k}^i$ which satisfies the time-varying input constraint $\mathcal{U}_k^i$, and the global input constraint $\mathcal{U}^i$ by definition of (6.26). Furthermore, $\tilde{u}_k^i$ robustly steers the local system into the terminal constraint $\mathbb{X}_\xi^i$ satisfying all local and global state constraints and time-varying input constraints for all possible deviations $\delta\tilde{\mathbf{u}}_k^i \in \delta\tilde{\mathbb{U}}_k^i$.
In all time-steps $k+l$ for $1 \leq l < H$, a feasible input $u_{k+l}^i = u_{k+l|k}^i$ is given by the tuple $(\tilde{v}_k^i, K_k^i)$ designed in k with:

$$u_{k+l}^i := v_{k+l|k}^i + \sum_{j\in\mathcal{N}^i} \sum_{m=k-a_k^{ij}+1}^{k+l-a_{k+l|k}^{ij}} \eta_{k+l,m|k}^{ij} \delta u_{m|k}^{ij}. \tag{6.39}$$

Note, that the lower and upper bounds follow from Prop. 1.1, and that the last required deviation is $\delta u_{k+l-a_{k+l|k}^{ij}|k}^{ij}$. Recall (6.18) for time-step $k+l$, and the fact that the last exactly known input from subsystem j in $k+l$ is given by $u_{k+l-a_{k+l}^{ij}}^j$. Then, the last known deviation is $\delta u_{k+l-a_{k+l}^{ij}|k}^{ij}$, which is at least as old as the desired one, if Asm. 3 holds with $a_{k+l|k}^{ij} \geq a_{k+l}^{ij}$. Through (6.30), input u_{k+l}^i satisfies local input constraints, if all $\delta u_{m,k}^{ij} \in \Delta\mathcal{U}_{m|k}^{ij}$, which is true, if all subsystems j update their local input trajectory constraint according to (6.38). Additionally, state constraints are also satisfied through (6.30).

In time-step $k+H$, an admissible input can be calculated according to the terminal control law $u_{k+H}^i = K_\xi^i \xi_{k+H|k}^i$, since the terminal state $\xi_{k+H|k}^i$ is robustly steered into $\mathbb{X}_\xi^i$ through constraint (6.30) within (6.23). If Asm. 4 holds and $\mathbb{X}_\xi^i$ is robustly invariant for any permissible $\mathbf{u}_{k+H|k}^i$, state and input constraints are satisfied.

Thus, the Asm.s 3 and 4 ensure the existence of the admissible input trajectory in $k+1$:

$$\tilde{u}_{k+1}^i := \left[u_{k+1|k}^{i\intercal}, \ldots, u_{k+H-1|k}^{i\intercal}, (K_\xi^i \xi_{k+H|k}^i)^\intercal \right]^\intercal. \tag{6.40}$$

Feasibility of (6.36) for all times $k + l \ \forall l \in \mathbb{N}_{>0}$, follows directly by induction over k. $\square$

According to the signal $\mathbf{u}^i_{k+l|k}$ in the cost function (6.31), stability of the distributed control system is proven with respect to the *ISpS*-property, which is stated and proven below:

Definition 2. *From [90] it is well known, that an autonomous system $z_{k+1} = f(z_k, w_k)$ with bounded disturbance $w_k \in \mathcal{W}$ is input-to-state practical stable (ISpS) in a forward positive invariant $\mathcal{Z}$, if there exist constants $d_1, d_2 \geq 0$, $a_1, a_2, a_3, a_e > 0$, and $\mathcal{K}$-functions $\alpha_1(r) = a_1 r^{a_e}, \alpha_2(r) = a_2 r^{a_e}, \alpha_3(r) = a_3 r^{a_e}$, and $\phi(r)$ respectively, and a function $V : \mathcal{Z} \to R_{\geq 0}$ such that for all $z \in \mathcal{Z}$:*

$$\alpha_1(\|z_k\|) \leq V(z_k) \leq \alpha_2(\|z_k\|) + d_1, \tag{6.41}$$

$$V(z_{k+1}) - V(z_k) \leq -\alpha_3(\|z_k\|) + \phi(\|w_k\|) + d_2 \tag{6.42}$$

hold for all $w_k \in \mathcal{W}$, where $\|z_k\|$ denotes any norm. As in [91], the relaxation of (6.42) to a multi-step definition is used with $\hat{H} \in \mathbb{N}_{\geq 1}$, such that:

$$V\left(z_{k+\hat{H}}\right) - V(z_k) \leq \alpha_3(\|z_k\|) + \hat{H}\phi\left(\left\|w_{[k,\hat{H}]}\right\|\right) + \hat{H}d_2 \tag{6.43}$$

needs to be proven with $\left\|w_{[k,\hat{H}]}\right\| = \max\limits_{r \in \{k,\dots,k+\hat{H}\}} \|w_r\|$.

Defining an extended state as the triple $z^i_k = (\mathbf{x}^i_k, v^i_k, \mathbf{u}^i_k)$, the norm and the trajectory are given by $\|z^i_k\|^r_{Q^i_z} := \|\mathbf{x}^i_k\|^r_{Q^i_x} + \left\|\left[v^i_k; \mathbf{u}^i_k\right]\right\|^r_{Q^i_u}$, and $\tilde{z}^i_k = (\tilde{\mathbf{x}}^i_k, \tilde{v}^i_k, \tilde{\mathbf{u}}^i_k)$ respectively. Note that the cost function (6.31) explicitly depends on $\tilde{z}^i_k$, but the value function (6.36) simply depends on the tuple $(\mathbf{x}^i_k, \tilde{\mathbf{u}}^i_k)$, since $\tilde{\mathbf{x}}^i_k$ follows from the auxiliary condition with initial state $\mathbf{x}^i_k$, and $\tilde{v}^i_k$ is an optimization variable. Thus, the condition $V^i_k(\mathbf{x}^i_k, \tilde{\mathbf{u}}^i_k) = J^i_k\left(\tilde{\mathbf{x}}^{i\star}_k, \tilde{v}^{i\star}_k, \tilde{\mathbf{u}}^i_k\right) = J^i_k\left(\tilde{z}^{i\star}_k\right) = V^i_k$ holds, where $\star$ denotes the solution of the optimization (6.36).

Theorem 5. *If Asm.s 3 and 4 hold, system* (6.17) *with control law* (6.25) *is ISpS, if the optimization problem* (6.36) *is feasible, and if weights Q^i_x and Q^i_u are chosen such that the set $\Omega^i = \{z | L^i(z) = 0\}$ is not empty.*

Proof. In order to establish the ISpS property, the set $\mathcal{Z}^i$ is chosen equal to the set in the triple $\tilde{z}^i_k$ for which (6.36) is feasible, and the set $\tilde{\Omega}^i = \Omega^i \times \cdots \times \Omega^i$ for the extended state trajectory.

To consider deviations from previously communicated predictions of neighbored subsystems, i.e. $\mathbf{u}^i_{k+l|k} \neq \mathbf{u}^i_{k+l|k+l}$, the difference between two predictions is defined as $\delta z^i_{k+l|k} = z^i_{k+l|k+1} - z^i_{k+l|k}$, and the difference between to predicted trajectories is $\delta \tilde{z}^i_k = \left[\delta z^{i\intercal}_{k+1|k}, \dots, \delta z^{i\intercal}_{k+H|k}\right]^\intercal$, with $\delta z^i_{k+H|k} = \left(\mathbf{x}^i_{k+H|k+1} - \mathbf{x}^i_{k+H|k}, 0, 0\right)$. Since cost function (6.31) is a quadratic function, which is zero if $\tilde{z}^i_k \in \tilde{\Omega}^i$ (according to Asm. 4.4), it is straightforward to see, that there exist lower and an upper comparison functions $\alpha_1\left(\|\tilde{z}^i_k\|_{\tilde{Q}^i_z}\right)$ and $\alpha_2\left(\|\tilde{z}^i_k\|_{\tilde{Q}^i_z}\right)$, such that (6.41) holds with $d_1 = 0$.

To proof the reduction of cost (6.43), consider the optimal cost in k if (6.36) is feasible:

$$V_k^i = \left\|\xi^i_{k+H|k}\right\|^2_{P^i_\xi} + \left\|\mathbf{x}^i_{k|k}\right\|^2_{Q^i_x} + \left\|\left[v^i_{k|k};\mathbf{u}^i_{k|k}\right]\right\|^2_{Q^i_u} \tag{6.44}$$

$$+ \sum_{l=0}^{H-2} \left\|\mathbf{x}^i_{k+l+1|k}\right\|^2_{Q^i_x} + \left\|\left[v^i_{k+l+1|k};\mathbf{u}^i_{k+l+1|k}\right]\right\|^2_{Q^i_u}. \tag{6.45}$$

Some general cost in $k+1$ are given by:

$$J^i_{k+1} = \left\|\xi^i_{k+H+1|k+1}\right\|^2_{P^i_\xi} + \left\|\mathbf{x}^i_{k+H|k+1}\right\|^2_{Q^i_x} \tag{6.46}$$

$$+ \left\|\left[v^i_{k+H|k+1};\mathbf{u}^i_{k+H|k+1}\right]\right\|^2_{Q^i_u} \tag{6.47}$$

$$+ \sum_{l=0}^{H-2} \left\|\mathbf{x}^i_{k+l+1|k+1}\right\|^2_{Q^i_x} + \left\|\left[v^i_{k+l+1|k+1};\mathbf{u}^i_{k+l+1|k+1}\right]\right\|^2_{Q^i_u}. \tag{6.48}$$

Now, first consider the case that there is no difference between predicted inputs of neighbored subsystems, i.e. $\mathbf{u}^i_{k+l|k+1} := \mathbf{u}^i_{k+l|k}$. Applying in $k+1$ the input sequence defined in (6.40), the result is a trajectory of admissible triples $\tilde{z}^i_{k+1}$ without difference in predictions, i.e. $\delta\tilde{z}^i_k = 0$. An upper bound for the optimal cost difference is given by:

$$V_{k+1}{}^i - V^i_k \leq \left\|\mathbf{x}^i_{k+H|k+1}\right\|^2_{Q^i_x} \tag{6.49}$$

$$+ \quad \left\|\left[v^i_{k+H|k+1};\mathbf{u}^i_{k+H|k+1}\right]\right\|^2_{Q^i_u} + \left\|\xi^i_{k+H+1|k+1}\right\|^2_{P^i_\xi} \tag{6.50}$$

$$- \quad \left\|\mathbf{x}^i_{k|k}\right\|^2_{Q^i_x} - \left\|\left[v^i_{k|k};\mathbf{u}^i_{k|k}\right]\right\|^2_{Q^i_u} - \left\|\xi^i_{k+H|k}\right\|^2_{P_\xi}. \tag{6.51}$$

Note the following three facts:

1. $\mathbf{x}^i_{k+H|k+1} := \mathbf{x}^i_{k+H|k}$, and $v^i_{k+H|k+1} := K^i_\xi \xi^i_{k+H|k}$, such that: $\left\|\left[K^i_\xi\xi^i_{k+H|k};\mathbf{u}^i_{k+H|k+1}\right]\right\|^2_{Q^i_u} \leq \left\|\left[K^i_\xi\xi^i_{k+H|k};\mathbf{0}\right]\right\|^2_{Q^i_u} + \left\|\left[\mathbf{0};\mathbf{u}^i_{k+H|k+1}\right]\right\|^2_{Q^i_u}$ and $\left\|\mathbf{x}^i_{k+H|k}\right\|^2_{Q^i_x} + \left\|\left[K^i_\xi\xi^i_{k+H|k};\mathbf{0}\right]\right\|^2_{Q^i_u} = \left\|\xi^i_{k+H|k}\right\|^2_{Q^i_\xi}$.

2. $\xi^i_{k+H+1|k+1} = A^i_\xi\xi^i_{k+H|k+1} + B^i_\xi\mathbf{u}_{k+H|k+1}$ with $\xi^i_{k+H|k+1} := \xi^i_{k+H|k}$ such that: $\left\|\xi^i_{k+H+1|k+1}\right\|^2_{P^i_\xi} \leq \left\|A^i_\xi\xi^i_{k+H|k}\right\|^2_{P^i_\xi} + \left\|B^i_\xi\mathbf{u}^i_{k+H|k+1}\right\|^2_{P^i_\xi}$.

3. P^i_ξ is assumed to solve the Lyapunov equation for the closed-loop system (6.33), such that: $\left\|A^i_\xi\xi^i_{k+H|k}\right\|^2_{P^i_\xi} - \left\|\xi^i_{k+H|k}\right\|^2_{P^i_\xi} + \left\|\xi^i_{k+H|k}\right\|^2_{Q^i_\xi} \leq 0.$

Considering these facts, an upper bound results according to Asm. 4:

$$V^i_{k+1} - V^i_k \leq - \left\|\mathbf{x}^i_{k|k}\right\|^2_{Q^i_x} - \left\|\left[v^i_{k|k};\mathbf{u}^i_{k|k}\right]\right\|^2_{Q^i_u} \tag{6.52}$$

$$+ \left\|\left[\mathbf{0};\mathbf{u}^i_{k+H|k+1}\right]\right\|^2_{Q^i_u} + \left\|B^i_\xi\mathbf{u}^i_{k+H|k+1}\right\|^2_{P^i_\xi}. \tag{6.53}$$

Since $\mathbf{u}^i_{k+H|k+1} \in \text{prod}(\mathcal{U}^j, j \in \mathcal{N}^i)$ is bounded, there exists a constant d_2 and a comparison function α_3 such that:

$$V^i_{k+1} - V^i_k \leq -\alpha_3\Big(\big\|z^i_k\big\|_{Q^i_z}\Big) + d_2. \tag{6.54}$$

Now consider the general case with possible deviations of previously communicated predictions, i.e. $\delta\tilde{z}^i_k \neq 0$. Analogously to $\delta\tilde{z}^i_k$, the difference of two predicted terminal states is denoted by $\delta\xi^i_{k+l|k}$. To the norm of possible deviations from above, the triangle inequality applied for all $l \in \{0, \ldots, H-2\}$ leads to:

$$\big\|z^i_{k+l+1|k+1}\big\| - \big\|z^i_{k+l+1|k}\big\| \leq \big\|\delta z^i_{k+l+1|k}\big\|, \tag{6.55}$$

and to the state in fact 1):

$$\big\|\mathbf{x}^i_{k+H|k+1}\big\|^2_{Q^i_x} \leq \big\|\mathbf{x}^i_{k+H|k}\big\|^2_{Q^i_x} + \big\|\delta z^i_{k+H|k}\big\|^2_{Q^i_z}, \tag{6.56}$$

and to the input in fact 1):

$$\big\|\big[K^i_\xi \xi^i_{k+H|k+1}; \mathbf{0}\big]\big\|_{Q^i_u} \leq \big\|\big[K^i_\xi \xi^i_{k+H|k}; \mathbf{0}\big]\big\|_{Q^i_u} + \big\|\big[K^i_\xi \delta\xi^i_{k+H|k}; \mathbf{0}\big]\big\|_{Q^i_u}, \tag{6.57}$$

and to the terminal state in fact 2):

$$\big\|A^i_\xi \xi^i_{k+H|k+1}\big\|^2_{P^i_\xi} \leq \big\|A^i_\xi \xi^i_{k+H|k}\big\|^2_{P^i_\xi} + \big\|A^i_\xi \delta\xi^i_{k+H|k}\big\|^2_{P^i_\xi}. \tag{6.58}$$

Then, the upper bound (6.54) is obtained to:

$$V^i_{k+1} - V^i_k \leq -\alpha_3\Big(\big\|z^i_k\big\|_{Q^i_z}\Big) + d_2 + \big\|\big[K^i_\xi \delta\xi^i_{k+H|k}; 0\big]\big\|_{Q^i_u} \tag{6.59}$$

$$+ \big\|A^i_\xi \delta\xi^i_{k+H|k}\big\|^2_{P^i_\xi} + \sum_{l=1}^{H} \big\|\delta z^i_{k+l|k}\big\|^2_{Q^i_z}. \tag{6.60}$$

Since a difference in the terminal state $\delta\xi^i_{k+H|k}$ comprises multiple differences in the augmented states $\mathbf{x}^i_{k+l|k}$, it strictly depends on the difference of predictions $\delta\tilde{z}_k$. Therefore, it is straightforward that there exists a comparison function ϕ, such that:

$$V^i_{k+1} - V^i_k \leq -\alpha_3\Big(\big\|z^i_k\big\|_{Q^i_z}\Big) + \phi\Big(\big\|\delta\tilde{z}^i_k\big\|_{\tilde{Q}^i_z}\Big) + d_2 \tag{6.61}$$

holds.

Since there is no guarantee to receive new information in $k+1$, and possible deviations of previously communicated predictions are not known before new data arrives, the scheme needs to by applied recursively. Assuming $\hat{H} = H \geq \bar{a}^i$, data will be known and the decrease of the value function is upper bounded by:

$$V^i_{k+H} - V^i_k \leq -\alpha_3\Big(\big\|\tilde{z}^i_k\big\|_{\tilde{Q}^i_z}\Big) + H\sigma\Big(\big\|\delta\tilde{z}^i_{[k:k+H]}\big\|_{\tilde{Q}^i_z}\Big) + Hd_2, \tag{6.62}$$

which complies to (6.43) and implies the ISpS property. □

Note that under certain conditions the stricter property of *input-to-state stability* (ISS) can be derived.

Corollary 1. *If system (6.17) with control law (6.25) is ISpS, it is also ISS if all subsystems $j \in \mathcal{N}$ converge to the origin of their respective terminal set.*

Proof. The terminal control law $u_k^i = K^i \mathbf{x}_{k-\bar{a}^i+1}^i$ steers all subsystems $i \in \mathcal{N}$ to the origin of their respective terminal set. If converged, the control law results with $u_k^i = 0$. If all subsystems converge to the origin at the end of their prediction horizon, this implies $\mathbf{u}_{k+H|k+1}^i = 0$, and since (6.54) - (6.62) d_2 depends exclusively on $\mathbf{u}_{k+H|k+1}^i$, $d_2 = 0$ follows. Together with $d_1 = 0$ (as discussed already in the proof to Theorem 2), the ISS property is implied. $\square$

6.2.5 Simulation Example

In this section, the presented framework is applied to the example of 3 autonomous vehicles (subsystems denoted by S1 to S3 hereafter) which move as a platoon. The common control objective consists of minimizing the distances between the vehicles, while avoiding collision. It is assumed that the vehicles communicate over wireless channels, and that the tasks of the centralized network controller are either performed by one of the vehicles itself, or by a road side unit. Either way, the platoon is organized in such a way that each subsystem (corresponding to an agent in the CNET) only requires data from its immediate predecessor. Hence, S1 acts autonomously, S2 receives data from S1 (with AoI $a_k^{2,1}$), and S3 receives data from S2 (with AoI $a_k^{3,2}$), as illustrated in Figure 6.4.

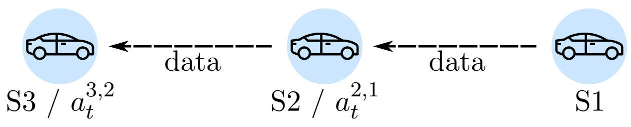


Fig. 6.4: Structure of the platoon example.

Given the goal of *robust* control of the platoon in the CSYS, the forecasts predicted by the network controller in the CNET have to be reliable according to Asm. 3. This is accomplished by 2 minor tweaks to the CNET model: First, the DTMC $\{s_k\}$ (which governs the quality of the links) is modeled with deterministic transition probabilities. In particular, each state of $\{s_t\}$ dictates the two mean transition probabilities $\bar{p}_k^2$ and $\bar{p}_k^3$ for the transmission between S1 $\rightarrow$ S2 and S2 $\rightarrow$ S3, respectively, as illustrated in Figure 6.5.

Secondly, the following minor deviation from the MPC paradigm for the network controller is considered: Instead of optimizing for the entire horizon without regard of the previously calculated optimal control trajectory, the network controller only optimizes for the last step of the horizon, while all prior controls are given by the previous trajectory. Though this constrains the control of the CNET, it enables the strict robustness result for the control of the CSYS from the previous section.

Doing so, the evolution of the AoI is shown in Figure 6.6 (top) for a random realization of the link transmissions variables p_k^1 and p_k^2. The blue-shaded area represents the actual AoI $a_k^{2,1}$ and $a_k^{3,2}$, that is at least 1 and at most 4. Additionally, the red-shaded area represents the predicted AoI. Not quite visible is the fact that the red-shaded bars always include the blue ones.

The local dynamics (6.14) of the subsystems S1, S2, and S3 are parameterized

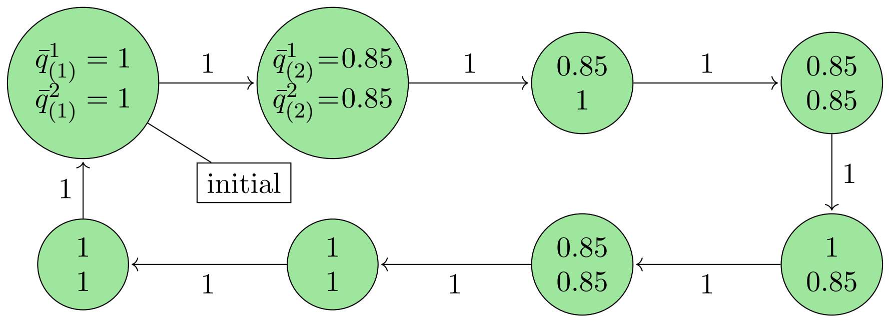


Fig. 6.5: DTMC $\{s_k\}$ governing the values of the mean transmission success probabilities by assigning $\bar{p}^1_k = \bar{q}^1_{(s_k)}$ (top entry) and $\bar{p}^2_k = \bar{q}^2_{(s_k)}$ (bottom entry).

identically and as in [54] with:

$$A^i = \begin{bmatrix} 1 & 0.3 \\ 0 & 1 \end{bmatrix}, \; B^i = \begin{bmatrix} 0.045 \\ 0.3 \end{bmatrix}, \; i \in \mathcal{N} = \{1, 2, 3\}. \tag{6.63}$$

The state $x^i_k = \begin{bmatrix} x^i_{1,k} \; x^i_{2,k} \end{bmatrix}^\intercal$ consists of the position $x^i_{1,k}$ and velocity $x^i_{2,k}$. The input is given by the acceleration u^i_k, constraint to $\|u^1_k\| \leq 1.98$, $\|u^2_k\| \leq 3$, and $\|u^3_k\| \leq 5$. The first vehicle S1 follows an internal reference, such that the focus is on the behavior of S2 and S3, with augmented states $\mathbf{x}^{i\intercal}_k = \begin{bmatrix} x^{i\intercal}_k \; x^{i-1\intercal}_k \end{bmatrix}, i \in \{2, 3\}$. The weights of the cost functions are chosen to $Q^i_x = \begin{bmatrix} Q & -Q \\ -Q & Q \end{bmatrix}$, with $Q = \mathrm{diag}(5, 1)$, and $Q^i_u = \begin{bmatrix} 0.1 & -0.1 \\ -0.1 & 0.1 \end{bmatrix}$ respectively. Note, that Q^i_x and Q^i_u are indefinite, but a decrease of stage cost in direction of all local states and inputs is still guaranteed. To avoid collisions, the augmented state is constrained to $0 \leq x^{i-1}_{1,k} - x^i_{1,k} \leq 200$. According to Figure 6.5, the maximum possible AoI is 4. Hence, the terminal control laws need to compensate 4 time-steps and are chosen to $u^2_k = \begin{bmatrix} -0.03 \; -0.54 \; 0.03 \; 0.54 \end{bmatrix} \mathbf{x}^2_{k-4}$, and $u^3_k = \begin{bmatrix} -0.06 \; -0.6 \; 0.06 \; 0.6 \end{bmatrix} \mathbf{x}^3_{k-4}$ to satisfy Asm. 4. The computation of the terminal sets follows according to the methods described in [92] and the model (6.33). With a control horizon of $H = 8$, the simulation starts at $k = 0$ with initial states $x^{ref}_0 = [0 \; 5]^\intercal$, $x^1_0 = [-13 \; 5]^\intercal$, $x^2_0 = [-20 \; 5]^\intercal$, and $x^3_0 = [-25 \; 5]^\intercal$, where ref denotes the internal reference of S1. This reference corresponds to accelerating with a constant value of $u^{ref}_k = 1$. The vehicles S2 and S3 minimize the distance to their predecessor, and S1 to its reference respectively.

In $k = 20$, an unexpected change of the setpoint of S1 is modeled to keep a constant distance of 5 meters to the reference. Thus, S1 suddenly decelerates and deviates from the prediction, which was communicated to S2 before (but remains inside the bounds predicted at $k = 19$). As soon as S2 receives the information of this deviation, it reacts with deviating from its own plan previously predicted and communicated to S3 (but again within the communicated bounds). This scheme repeats for S3. Figure 6.6 shows the distance of all subsystems with respect to the reference of S1, where the trajectory of the last subsystem is shown as dashed line and the trajectory of the first subsystem as a blue solid line. The control results differ with respect to the use of the forecasted age of information (fc) compared to the use of the worst-case age (wc). To quantify the conservatism of both simulations, the *ideal* behavior – which would result if the communication network could a priori predict the true values of the AoI – is shown, too. At $k = 60$, a significant delay

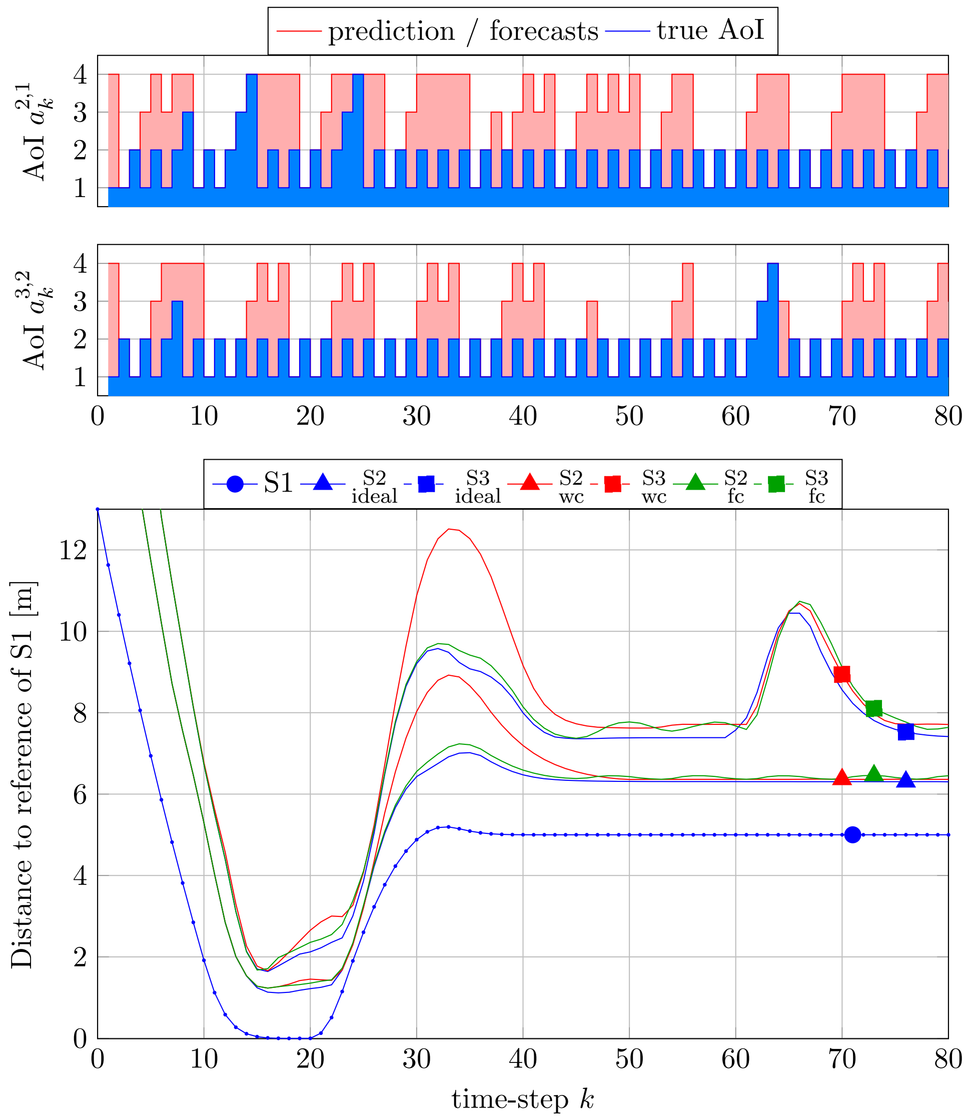


Fig. 6.6: Predicted and actual values for the AoI (top), and distance of vehicles to the reference of S1 with unpredicted setpoint change of S1 at $k = 20$ with use of the forecasts (fc) in contrast to the ideal behavior (ideal) and the use of a worst-case delay (wc) (bottom).

of sending information from S2 to S3 in the communication network is modeled (see Figure 6.6 top), such that S3 has to decelerate and to keep more distance to S2. This results in the same behavior according to the use of the worst-case age of information. Prior delays in the communication network have lower effect caused by general transient effects of the distributed plant.

The figure shows that the maximum distance of the platoon can be reduced significantly for S2 and S3 by using the forecasts within the predictive vehicle controllers. For the considered simulation, the distance of the platoon (measured from S1 to S3) caused by the setpoint change of S1 is reduced by over 37%, from 7.5 m to 4.7 m, still avoiding collisions in a robust way. This improvement of performance mainly results from the reduced predicted age of information, and the less restrictive constraints constructed thereon in the subsystem controllers. The comparison to the *ideal* behavior unveils that the amount of conservatism introduced by the uncertain communication almost vanishes if the AoI forecasts are used.

6.2.6 Conclusions

A new control scheme for a class of Cyber-Physical System has been presented, which interweaves the control strategy of the communication system with the control strategy of the distributed plant. As a main result, robust stability in the sense of ISpS has been proven for the MPC scheme of the distributed plant, given the assumption that the predictive network controller provides reliable forecasts of the age of information. The principle of controlling the communication network by a predictive control scheme offers the possibility to generate such forecasts (as a by-product) with adjustable reliability. The inclusion of the forecasts decreases the conservatism of the robust DMPC scheme for plant control. The amount of improvement is difficult to quantify in a general sense, but strongly depends on the difference of the predicted delay to the upper bound of possible delays: the larger the average difference over the operation is, the more significant the performance increase will be.

The presented methodology is relevant for all applications for which a controlled distributed plant interacts through a communication network with time-varying communication delay. With respect to 5G communication, this applies to many Cyber-Physical Systems, including all autonomous applications of driving and robotics.

Future work will extend the considerations to a framework in which the distributed plant control will consider the age of information in stochastic representation. This appears as a viable alternative if the computation of the robust positive invariant sets is numerically challenging, as observed for higher-dimensional dynamics.

6.3 Concluding Remarks

The relaxations, proposed to ease the computational burden of the optimization (6.5) are heuristic and the paper does neither mention nor quantify their impact on the quality of the generated solution (compared to the true optimal solution of (6.5)). To rectify, we present the results of a Monte-Carlo simulation, based on a set-up consisting of 5 agents where each agent requires status information of all other agents (i.e. all weights in (6.5) are set to 1). The prediction horizon is 5 and there are 5 Markov-states, each with a separate set of transmission success probabilities for each possible link between the agents. In each instance of the Monte-Carlo simulation, these sets of transmission success probabilities (identifiable via a symmetric matrix of dimension 5×5) as well as the transition matrix for the DTMC are rerolled. Using less agents or a smaller prediction horizon does simplify the optimization and therefore, most likely, would result in sub-optimal solutions (produced by the relaxed optimization) which are closer to the actual optimal solution, compared to the chosen set-up. On the other hand, using more agents or a larger prediction horizon does exceed our available computational resources.

To evaluate the quality of the solution of the relaxed optimization, we calculate its relative distance to the true optimal solution, labeled ρ. Let x_{best} and x_{worst} be the true optimal solution and the worst solution (i.e. the solution that maximizes the objective instead of minimizing it), respectively. Then, with x being the (sup-optimal) solution of the relaxed optimization, the relative distance becomes

$$\varepsilon = \frac{x - x_{\text{best}}}{x_{\text{worst}} - x_{\text{best}}} \tag{6.64}$$

The results of 1000 simulation runs are illustrated in Figure 6.7, where a cumulative distribution function of ρ is plotted. Through all our runs, the largest deviation from the true optimal solution was less than 15% while in around 20% of all cases, there was no difference at all. This justifies the relaxations made in the paper.

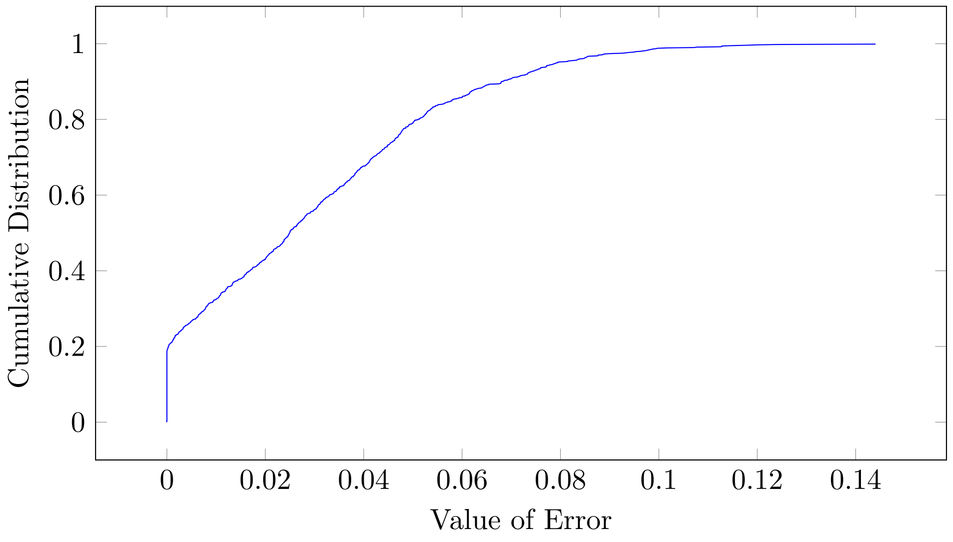


Fig. 6.7: Cumulative Distribution of the error made by the relaxed optimization. In about 20% of all cases, there is no difference between true and relaxed optimization. In 90% of all cases, the error is well beyond 7%.

Paper 7

State-Space of AoI

Full Paper Title An Algorithm for Exact Numerical Age-of-Information Evaluation in Multi-Agent Systems

Publisher IEEE International Conference on Communications (ICC), 2022

Authors Richard Schöffauer, Gerhard Wunder

Acknowledgment Richard Schöffauer is supported by the DFG under priority programm SPP 1914. Gerhard Wunder is supported by both the DFG (SPP 1914) and the BMBF (6G-RIC).

Abstract We present an algorithm for the numerical evaluation of the state-space distribution of an Age-of-Information network. Given enough computational resources, the evaluation can be performed to an arbitrary high precision. An Age-of-Information network is described by a vector of natural numbers, that track how outdated status information from various agents is. Our algorithm yields the means to determine any moment of the corresponding stochastic process. This can be extremely valuable for cases in which the network consists of controllers that communicate with one another, as it potentially allows for less conservative control behavior. It also enables the comparison of different policies regarding their performance (minimizing the average Age-of-Information) to a much more accurate degree than was possible before. This is illustrated using the conventional MaxWeight policy and the optimal policy. We also validate and compare the algorithm with Monte-Carlo simulations.

7.1 Preliminary Remarks

As seen in the previous paper, optimizing the AoI via scheduling decisions is quite intricate, especially if the optimization is supposed to hold for an entire prediction horizon. Naturally the question arises whether there exists an optimal scheduling policy, i.e. a policy that guarantees a minimum AoI, averaged over all agents and over all time. To the best of our knowledge such a policy does not exist in analytical form, though, given the discreteness of the problem, it is possible to derive it in a tabular description. This is possible through the means of dynamic programming [93] which provides a set of iterative methods that run over the state-space and in each iteration improve on a value function and the policy. Caused by the iterative nature of these methods, the true value function (which can be formulated such as to yield the expected AoI as a function of the system-state) is only approached in the limit. In contrast, the paper at hand presents a method which yields the exact probability distribution over the state-space under any policy in tabular description. This automatically means that the exact average AoI can be obtained together with any higher moments like e.g. the variance.

Note that though dynamic programming only yields the true value function of a given policy in the limit, the optimal policy itself is obtained already with a finite number of iterations, since the policy is basically a discrete function.

7.2 Paper Body

7.2.1 Introduction

In recent years, the so-called Age-of-Information (AoI) metric has gained considerable attention in lieu of the conventional Communication-Delay (ComDelay) metric. While ComDelay captures the elapsed time between transmission and reception, AoI measures the age of information at the receiver. To appreciate the difference, imagine a communication line with a 1 second ComDelay and assume that a packet of information is send over that line only every full hour. The ComDelay is not influenced by this usage and remains 1 second; however the AoI at the receiver is about 30 minutes on average. It starts from 1 second every full hour and peaks at 1 hour and 1 second just before the next reception.

In contrast to ComDelay, AoI is especially well suited towards the needs of the information-receiving agents in a network. And in cases where the receiving agent is a controller awaiting new status-updates on its input variables (i.e. when the network closes a control loop), AoI is the much more significant indicator for the overall performance. While AoI grows linearly in time, it is reset abruptly to one (or zero, depending on the context), once a new status-update is received. This non-linear behavior is the reason why analytical results on the stochastic process describing the AoI are rare and usually of limited quality.

In search for an optimal scheduling policy, i.e. a policy that minimizes the average AoI, [94] shows that a greedy policy is optimal, but only in the case of symmetric networks. For the general case, [95] derives requirements on the optimal policy by modeling the network as a Markov-Decision-Process. [88] shows, that the same approach allows to employ restless bandit methods for leveraging a well performing policy. In [89, 96, 97], the authors develop a lower bound on the average AoI and manage to link the performance of multiple low-complexity policies to that bound. However, the results are quite weak; e.g. while the MaxWeight policy seems to yield an average AoI very close to this lower bound in simulations, analytically it can only be guaranteed to yield less than double that bound.

AoI in networks with explicit multi-hop propagation is investigated in [98, 99], where the focus lies on specific topologies (e.g. line or ring networks). For a general topology, fundamental bounds on average and peak AoI are derived in [100].

As mentioned, AoI is especially well suited if the network is part of a closed control loop. In [101], the authors show how forecasting AoI values can improve the performance of model-predictive controllers with a common control goal. For cellular networked control systems, [102] compares the AoI metric with a Value-of-Information metric which considers the expected information content of an update. The most practical investigation is performed in [86], where the authors even consider processing delay and physical execution times for industrial wireless sensor-actuator networks, in order to derive policies that minimize the average AoI.

Our **contribution** makes it possible to derive an exact numerical ratio between the performance of the optimal (average AoI minimizing) policy and the MaxWeight policy in the case of a two-agent network. In addition to that, a straightforward extension of the presented algorithm allows to yield such results for any policy and any number of agents (only limited by computational resources). Since the algorithm computes the exact probability distribution over the state-space, it enables system designers to derive the stochastical moments of the network process, facilitating a more accurate prediction of the network performance (and in cases where the network is part of a control loop a less conservative parametrization of the distributed controllers). In this paper, we present the algorithm including its application to the MaxWeight and the optimal policy.

7.2.2 System Model

The underlying system-model for this paper is inspired by the use case of distributed controllers, communicating in order to achieve a common control goal. Each controller operates autonomously and thus naturally generates status-updates of all relevant local quantities. Its individual control goal, however, is influenced by other quantities, only observable (because local) to other controllers in the network. Outdated information of those quantities directly diminishes the controller's performance. Hence, each controller is *always* motivated to broadcast its freshly generated status to all other controllers.

In particular, we assume a network of $n \in \mathbb{N}$ agents who communicate over a wireless resource such that only one agent may broadcast at any given time. (Here and throughout the paper, $\mathbb{N}$ does *not* contain $\{0\}$!). Time is slotted and in each slot, every agent generates an update of its own status that can potentially be broadcasted to all other agents. Every agent remembers the latest received status information of all other agents. Let the AoI of agent's i last broadcasted status-update at all other agents be denoted by $a_t^i \in \mathbb{N}$. The collection $(a_t^1, a_t^2, \dots a_t^n) \in \mathbb{N}^n$ therefore fully describes the network state in time-step t. Given no updates, a_t^i increases linearly with time but is *reset* to one, once a status-update is received. In order for this to happen, in each time-step a policy determines which agent is to broadcast its current status to all other agents and thus which AoI component is to be reset. If agent i is to broadcast in time-step t then the control variable $v_t^i \in \{0,1\}$ is one, otherwise it is zero. The resource constraint requires that $\sum_i v_t^i = 1$. Whether such a transmission from agent i to all other agents would succeed is determined by the stochastic variable $p_t^i \in \{0,1\}$, which is one in case of success and zero otherwise. The sequence $\{p_t^i\}$ describes a Bernoulli process with success probability $p^i \in [0,1] \subset \mathbb{R}$ (failure probability $\bar{p}^i = 1-p^i$) and serves to model channel fading and other stochastic disturbances. Put together this results in the following evolution of the AoI:

$$a_{t+1}^i = 1 + a_t^i \left(1 - v_t^i p_t^i\right) \tag{7.1}$$

With with slight abuse of notation, we let $a \in \mathbb{N}^n$ denote a state of the network and a^i $(i \in \{1, \dots n\})$ its components, independent of time. Since the system dynamics are time-invariant, we need only consider stationary policies that map each state a to a network agent i. Such a mapping, called a decision d fully defines a policy:

$$d : \mathbb{N}^n \to \{1, \dots n\} \tag{7.2}$$

Furthermore, we will only consider causal policies, i.e. policies for which holds

$$d(a) = i \quad \Longrightarrow \quad d(a + e^i) = i \tag{7.3}$$

with e^i being the i-th unit vector. I.e. given a state a and the policy's decision $d(a) = i$, the policy's decision stays on agent i, if we were to increase the AoI only in the i-th component. Most reasonable policies (like MaxWeight) fulfill this property.

7.2.3 Methodology for Evaluation of Exact State-Space Distribution

Notation & Technique

Let $f(a)$ be the probability, with which the network is in state a, given an arbitrary time-slot and no prior information. To evaluate $f(a)$ we utilize the special structure of the system evolution (7.1): Every state a, *not* at the boundary of the state-space can only be reached from its diagonal predecessor $a - \mathbf{1} = (a^1 - 1, \dots\ a^n - 1)$, and only if in this predecessor-state, the intended reset did fail (which happens with probability $\bar{p}^{d(a-\mathbf{1})}$). The sequence of consecutive predecessors of such a state form a diagonal line that ends at the state

$$a - \mathbf{1} \cdot \left(\min_i\{a^i\} + 1\right) = \left(a^1 - \min_i\{a^i\} + 1, \dots\ a^n - \min_i\{a^i\} + 1\right) \tag{7.4}$$

at the boundary of the state-space (dark-green states in Figure 7.1). We call this the "root-state" of a. Once the probabilities of all root-states are known, the remaining distribution over the state-space follows immediately by construction of said diagonals.

The policy determines how probability diminishes along a diagonal: The product of all failure probabilities $D(a)$, necessary to reach state a from its root-state, can be expressed recursively by

$$D(a) = \begin{cases} 1 & \text{if } \min_i\{a^i\} = 1 \\ D(a - \mathbf{1}) \cdot \bar{p}^{d(a-\mathbf{1})} & \text{otherwise} \end{cases} \tag{7.5}$$

Hence, we have the following connection between a state's probability and its root probability:

$$f(a) = D(a)f(a - \mathbf{1} \cdot \left(\min_i\{a^i\} + 1\right)) \tag{7.6}$$

Finally, given any causal policy, a root-state, let's say $(1, a^2, \dots\ a^n)$, can be expressed by

$$f(1, a^2, \dots\ a^n) = p^1 \sum_{a^1=b}^{\infty} f(a^1, a^2 - 1, \dots\ a^n - 1) \tag{7.7}$$

where b is determined by the employed policy. In particular, we say that state $(b, a^2 - 1, \dots\ a^n - 1)$ is the "first" state, from which $(1, a^2, a^3, \dots\ a^n)$ is reachable. A visual representation of (7.7) are the light-blue and light-green sets of states in Figure 7.1.

Evaluation of the State-Space Distribution under the MaxWeight Policy for $n = 2$

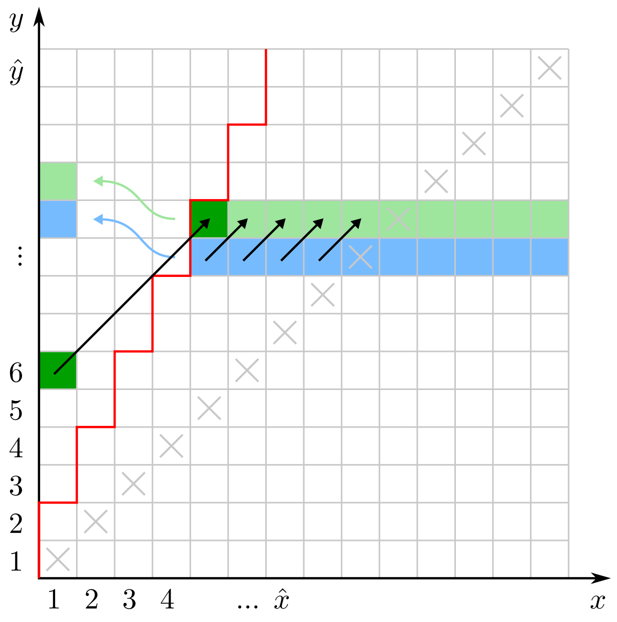


Fig. 7.1: Root-state recursion under MW policy in 2-dim. state-space. States on the boundary (root-states) are only reachable from the highlighted sets of states (light-green/blue). States not on the boundary are only reachable from their diagonal predecessor.

Given these tools, we will now investigate the AoI under the well known MaxWeight (MW) policy, in the 2-dimensional case. The methodology is applicable to higher-dimensional cases, albeit with slightly more linear algebra in order to determine the required variables. For succinct notation we will identify a^1 and a^2 with x and y, and p^1 and p^2 with p and q, respectively. Also, b from (7.7) will be denoted by $x'(y)$ or $y'(x)$, referring to the x or y component of the first state, from which the root-state in question is reachable. We can specify these functions given the employed policy: MW separates the state-space linearly into two halves. Whenever $xp \geq yq$, the system will try to reset x with probability p, otherwise y with

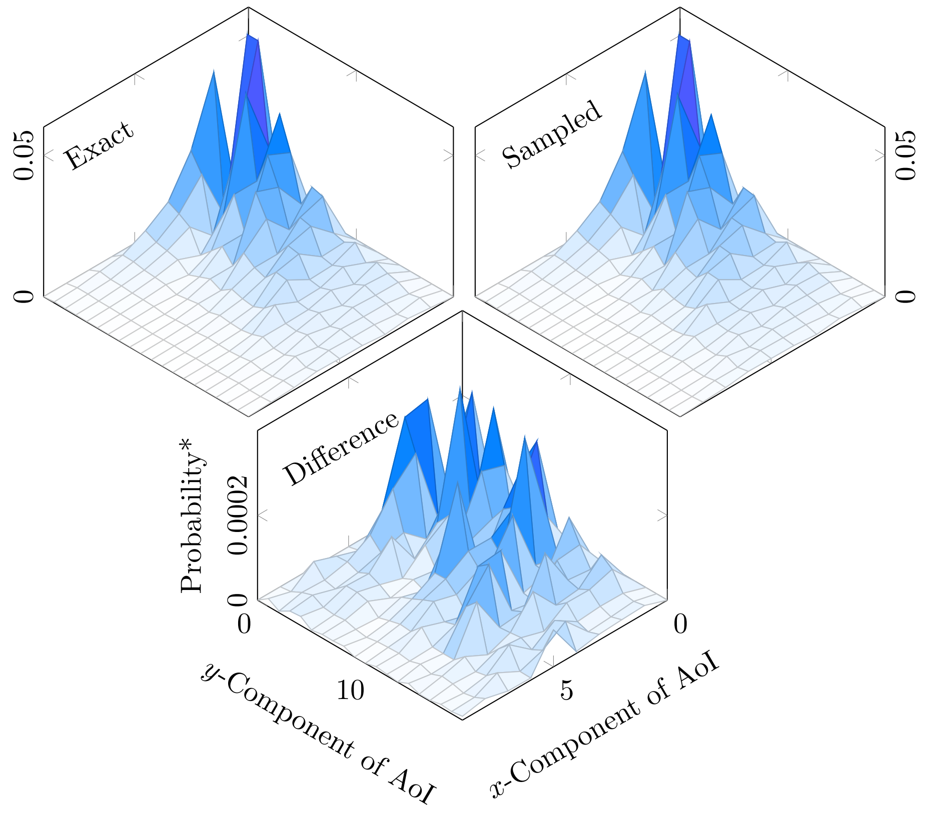


Fig. 7.2: Probability distribution over the state-space for the MW policy. Depicted are the exact values following the described method (upper left), the sampled values through 10^6 simulation steps (upper right) and the absolute difference between the two (down mid).

probability q. Hence the first state from which $(1, y)$ is reachable is $(x'(y-1), y-1)$ with

$$x'(y) = \left\lceil \frac{q}{p} y \right\rceil \tag{7.8}$$

since $x'(y)p \geq yq$; and vice versa for root-states of the form $(x, 1)$.

W.l.o.g. we will assume that $p > q$. Then, as visualized by Figure 7.1 (blue states), we can express the probability of a root-state on the y-axis as

$$\begin{aligned} f(1, y+1) &= p \sum_{x=x'(y)}^{\infty} f(x, y) \\ &= p \sum_{x=x'(y)}^{y} f(x, y) && + p \sum_{x=y+1}^{\infty} f(x, y) \\ &= p \sum_{x=x'(y)}^{y} f(1, y-x+1) D(x, y) && + p \sum_{x=y+1}^{\infty} f(x-y+1, 1) D(x, y) \end{aligned} \tag{7.9}$$

Crucially, no matter the choice of y, the last sum will always originate from the probability of all the root-states on the x-axis. In contrast, the first sum refers back to root-states on the y-axis that came before the state $(1, y)$, constituting a recursion.

To quickly yield this recursion, we compare (7.9) for states $(1, y+1)$ and $(1, y)$: The probability of reaching $(1, y+1)$ is equal to the probability of reaching $(1, y)$, attenuated by the probability that reaching $(1, y)$ did fail (diagonal shift from blue to green states in Figure 7.1). However, from time to time, $(1, y+1)$ can also be reached from an additional state whose predecessor does not stem from the states that $(1, y)$ was reachable from (dark green state in Figure 7.1). This "disturbance" in the recursion is caused by a change in $x'(y)$ and thus directly connected to the employed policy. The root of these additional states must be on the y-axis.

In particular, If $x'(y)$ is not $x'(y-1)+1$, there must be an additional state from which $(1, y+1)$ is reachable, namely $(x'(y), y)$. The root of this state is $(1, y - x'(y) + 1)$, and according to (7.6) we have

$$\begin{aligned} f(x'(y), y) &= f(1, y - x'(y) + 1) D(x'(y), y) \\ &= f(1, y - x'(y) + 1) \bar{q}^{x'(y)-1} \end{aligned} \tag{7.10}$$

This leads to the recursive formula

$$f(1, y+1) = \begin{cases} \bar{p} f(1, y) + p f(1, y - x'(y) + 1) \bar{q}^{x'(y)-1} & \text{if } x'(y) = x'(y-1) \\ \bar{p} f(1, y) & \text{otherwise} \end{cases} \tag{7.11}$$

With that we are just missing an initial value to start the recursion. Naturally we start with

$$f_A(1, 2) = p \sum_{x=x'(1)}^{\infty} f_A(x, 1) = 1 \tag{7.12}$$

since $f(1,1) = f(2,2) = \cdots = 0$ (x and y cannot be reset at the same time). We use f_A instead of f because we will have to scale the results in the end to ensure their sum equals 1. All formulas so far also hold true for f_A and we have $f = A \cdot f_A$ for some constant $A \in \mathbb{R}$ to be found. (In the multi-dimensional case, more initial values need to be used and thus more constants need to be considered.)

Henceforward, the entire distribution over the y-axis can be evaluated using (7.11). Evaluation must stop at some $\hat{y}$ due to practical limitations. Using the propagation along the diagonals via (7.6), the entire $(y > x)$-half of the state-space can be evaluated as well. Once this is done, the boundary distribution on the x-axis, $f_A(x, 1)$, follows as

$$f_A(x+1, 1) = q \sum_{y=y'(x)}^{\hat{y}} f_A(x, y) \tag{7.13}$$

Evaluation of the boundary distribution $f_A(x, 1)$ has to end at $\hat{x} = \left\lfloor \frac{q\hat{y}}{p} \right\rfloor + 1 \leq \hat{y}$. This is the x-component of the last state that is reachable from any prior evaluated states. Subsequently, the probability of the missing diagonals can be determined, after which f follows from f_A through normalization to 1 (i.e. finding A). Figure 7.2 shows the resulting distribution for the parameters $p = 0.6$ and $q = 0.2$.

Remark: For $\frac{p}{q} =: \kappa \in \mathbb{N}$ and $y = \kappa m + 1$ and $m = 1, 2, \ldots$ (i.e. looking only at every κ-th value of y), (7.11) simplifies to

$$f(1, \kappa m + 1) = \bar{p} f(1, \kappa m) + p \bar{q}^{m-1} f(1, (\kappa - 1)m + 1) \tag{7.14}$$

Every y value, not covered by this recursion can be expressed by exponential attenuation. A solution for this difference equation would allow for an analytical description of the distribution over the entire state-space. However, such a difference equation with *proportional delay* is notoriously hard to solve, especially since (7.14) exhibits a non-constant factor in front of the delayed term. The most recent results on similar equations can be found in [103] and references therein. In further abstraction, (7.14) is the discrete version of the so-called Pantograph equation, first published in [104]. However, due to the discrete argument, the difference equation does only hold for certain arguments (those which abide to the corresponding divisibility) whereas the Pantograph equation holds for all its real arguments. Therefore, most results on the Pantograph equation are not transferable to our problem.

Evaluation of the State-Space Distribution under the Optimal Policy for $n = 2$

Still staying in 2 dimensions, we will now discuss a more general example in which the policy is defined only numerically, i.e. $d(a)$ is given by a matrix. Once again we will identify a^1 and a^2 with x and y, and p^1 and p^2 with p and q, respectively. The policy shall now be defined by Figure 7.3: For all states above the separating line (red line), the policy tries to reset the y component, whereas for all states beneath, it tries to reset the x component.

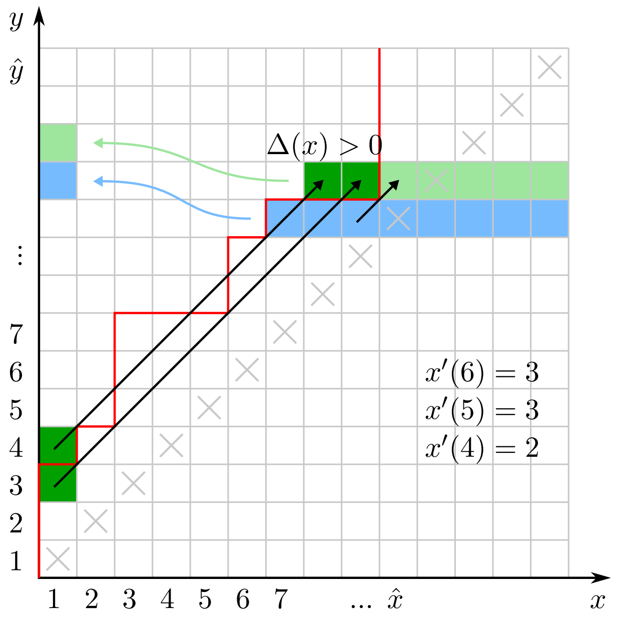


Fig. 7.3: General root-state recursion in 2-dim. state-space. The dark-green states are dropped compared to the previous line (blue). Their contribution has to be subtracted which requires involvement of their root states (also dark-green).

Since the function $x'(y)$ is now not given by a mathematical expression, we need to consider the change in $x'(y)$ for each step separately. Given the pure exponential attenuation of the root-states' probability in the case of $x'(y) = x'(y-1) + 1$, we define

$$\Delta(y) = x'(y) - x'(y-1) - 1 \tag{7.15}$$

With that, the following more general recursion holds

$$f(y+1,1) = \begin{cases} \bar{p}f(y,1) + p \sum_{i=0}^{-\Delta(y)-1} f(y,x'(y)+i) & \text{if } \Delta(y) < 0 \\ 0 & \text{if } \Delta(y) = 0 \\ \bar{p}f(y,1) - p \sum_{i=1}^{\Delta(y)} f(y,x'(y)-i) & \text{if } \Delta(y) > 0 \end{cases} \tag{7.16}$$

where root-states can be substituted via

$$f(y,x'(y) \pm i) = f(y - x'(y) \mp i + 1, 1) D(y, x'(y) \pm i) \tag{7.17}$$

With an initial parameter, the state-space can thus be evaluated using Algorithm 1.

Algorithm 1 Pseudo-Algorithm for Two-Agent System

```
Require: $f$ (2D-Matrix)
Require: $\hat{y}$ (Size of $y$-dimension of matrix $f$)
  $f(1,2) \leftarrow 1$
  $y \leftarrow 2$
  while $y < \hat{y} - 1$ do
    Evaluate $\Delta(y)$, using (7.15)
    Evaluate $f(1, y+1)$ using (7.16)
    Evaluate states, diagonal to $f(1, y+1)$ using (7.6)
    $y \leftarrow y + 1$
  end while
  $x \leftarrow 2$
  while $x \leq \lfloor \frac{q}{p} y \rfloor + 1$ do
    Evaluate $f(x, 1)$, using (7.13)
    $x \leftarrow x + 1$
  end while
  Normalize matrix $f$
```

We can employ the presented method on the *optimal* control policy (OP). Though to the best of our knowledge, an analytical expression is not yet found for this optimal policy, it is easily obtained by policy-iteration methods in form of a decision matrix like the one in Figure 7.3 (the separating line corresponds to the policy). For the case of $p = 0.9$ and $q = 0.1$, the decision matrices of the MW policy and the OP policy are depicted in Figure 7.4. The change in the resulting state-space distributions is visualized in Figure 7.5.

Doing this evaluation for every combination of $p = \{0.1, \ldots 0.9\}$ and $q = \{0.1 \ldots 0.9\}$, allows one to yield the comparison in performance of MW and OP policy. Therefor, every state probability is multiplied with its corresponding AoI value (identical to its L^0-norm). The sum over all states then gives the average AoI under the policy. As can be seen in Figure 7.6, OP has a performance gain (less average AoI) of about 15% over MW in the case of $p = 0.9$ and $q = 0.1$. The closer the probabilities are, the smaller this advantage gets, since the policies coincide for $p = q$.

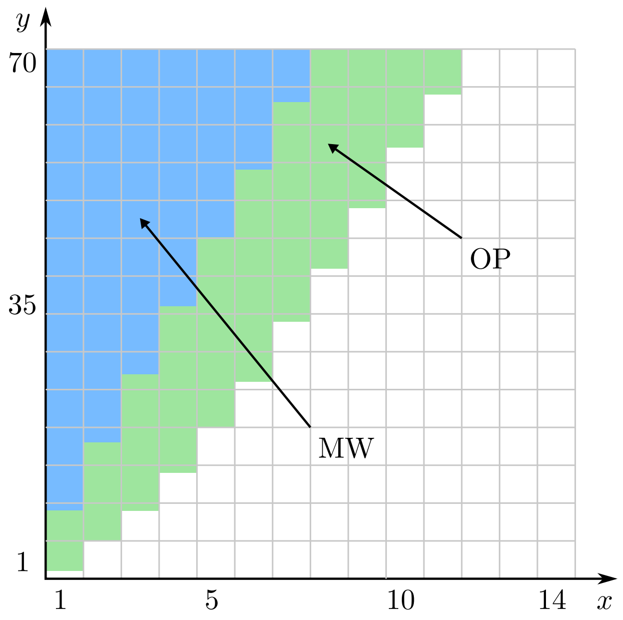


Fig. 7.4: Truncated decision matrix for MW (blue) and OP (blue + green). If the current state resides in the colored region, each policy tries to reset the y-component of the AoI with success probability $q = 0.1$, otherwise the x-component with probability $p = 0.9$. Compared with MW, OP tries to reset the y-component (corresponding to the weak communication link) much more often.

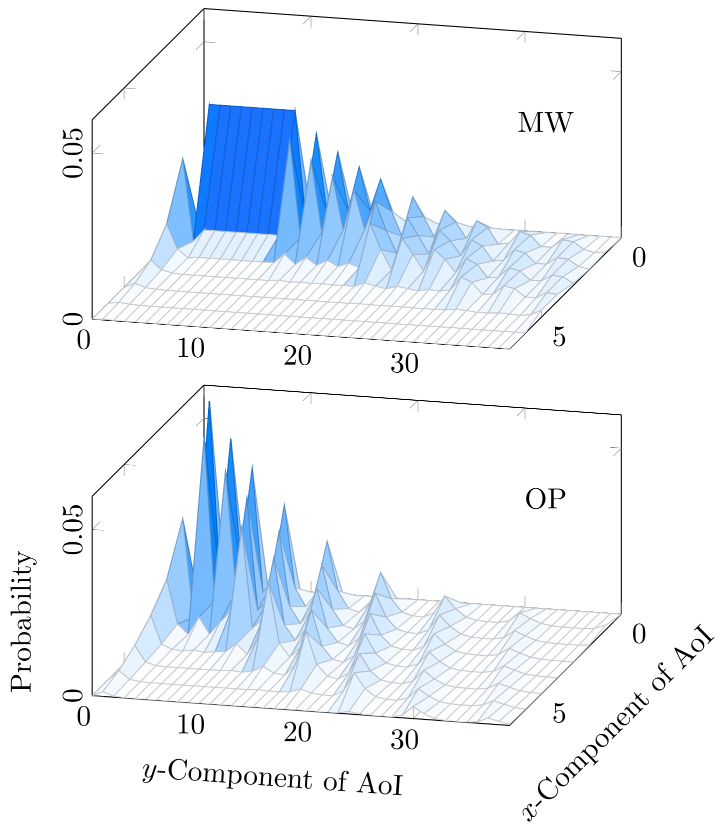


Fig. 7.5: State-space probability distribution for transmission success probabilities $p = 0.9$ and $q = 0.1$ under MW (top) and OP policy (bottom). The wave-like distribution under MW can be attributed to equation (7.14). An intuitive explanation is derived in Figure 7.8.

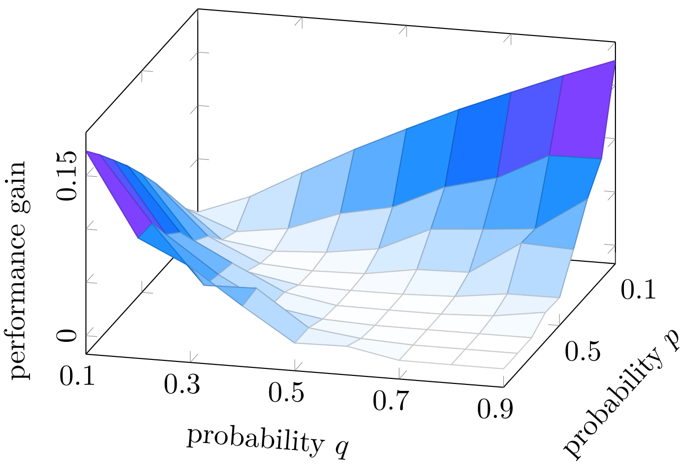


Fig. 7.6: Increased performance of OP over MW in % of average AoI drop. Probabilities p and q represent transmission success probabilities between the agents. For $p = q$, MW coincides with OP. For differing values of p and q, OP performs better.

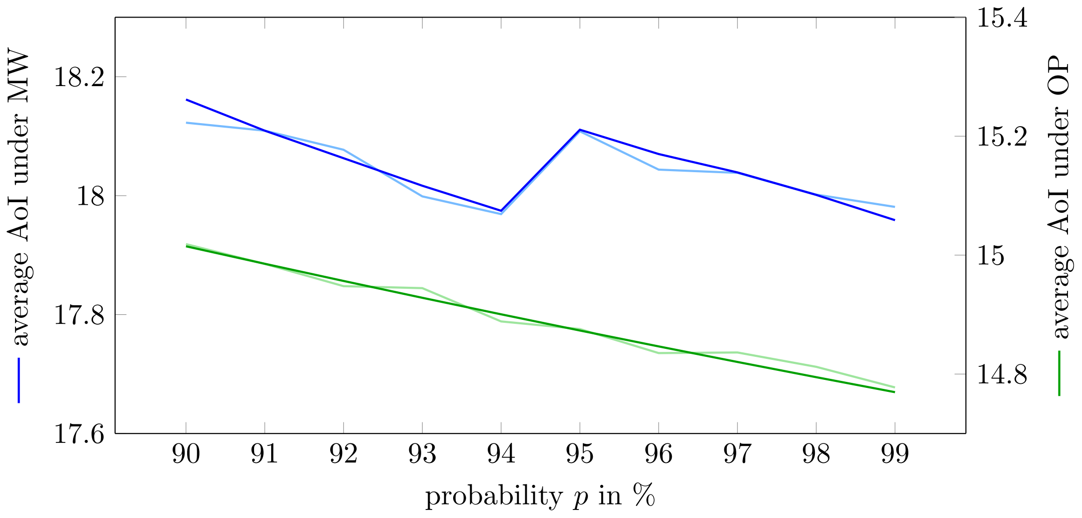


Fig. 7.7: Average AoI for $q = 0.1$ and $p = 0.9, \ldots 0.99$. Exact values using the algorithm (dark-blue/dark-green) and values obtained with Monte-Carlo simulation (10^7 steps) (light-blue and light-green). The non-smooth decay under MW stands out.

7.2.4 Noteworthy Observations

Looking at the results, there is a surprising observation visualized in Figure 7.7. As evaluated by the proposed algorithm and cross checked with Monte-Carlo simulations, there does not seem to be a smooth decay of the average AoI when increasing the success probability p under the MW policy. Setting $q = 0.1$ and looking at the values for $p = 0.94$ and $p = 0.95$, the average AoI increases while p increases. I.e. the system performance decreases though the network reliability strictly increases. Using only Monte-Carlo simulations to determine the average AoI, one would probably misinterpret such a finding as sample noise. Our algorithm, however, proves that this seems to be a genuine effect.

An explanation for this effect might be found in Figure 7.4. The figure shows that MW gives much less priority to the activation of the weak link (y-component of the AoI), compared to OP. Increasing p from $p = 0.94$ to $p = 0.95$ will lessen MW's priority on activating the weak link even more (since the weight of the strong link becomes even greater). Given that the state-space is discrete, the slight change in p will make some of MW's state-dependent decisions flip from "activate weak link" to "activate strong link" (equal to slightly less blue area in Figure 7.4). These non-smooth flips, together with the fact that for optimal behavior, a policy should rather try to activate the weak link more often might cause the observed behavior.

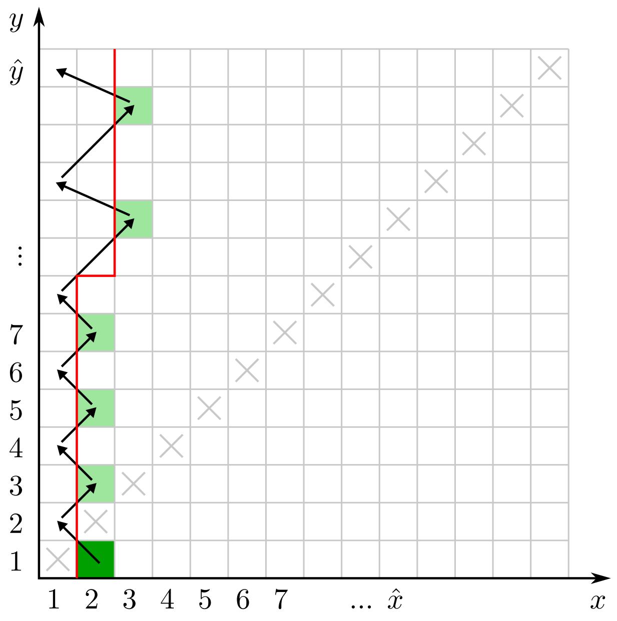


Fig. 7.8: Intuitive explanation for wave-like form of the probability distribution in Figure 7.5: Let the probability of a successful reset of x and y be high and low, respectively. Then, if the dark-green state is highly probable, so are the light-green states. And even if a y reset is to succeed during the indicated evolution, there is a high probability of reaching the dark-green state again.

7.2.5 Conclusion

We presented an algorithm with which, for the first time, it is possible to yield an exact numerical comparison between the AoI-optimizing policy and the MaxWeight policy in terms of average AoI in a two-agent network. For this case, we show the connection to a difference equation with proportional delay (related to the infamous Pantograph equation) and present simulation results for different transmission parameters. Our algorithm is based on the evaluation of the probability distribution over the state-space, facilitated by the special properties of the AoI process. It is readily applicable to any causal policy and can also be extended to the case of arbitrary many agents. In contrast to Monte-Carlo simulations, our algorithm allows for a much faster evaluation and additionally is much more exact, facilitating the evaluation of higher order moments of the underlying stochastic AoI process.

7.3 Concluding Remarks

Though it is true, that the method presented in this paper can be extended to more dimensions, it already requires an immense amount of effort only to denote the recurrence formulas for the case of 3 dimensions (i.e. 3 agents). The difficulty comes from the structure of the separation of the state-space, which is generated by a policy's mapping from a state-vector onto a control decision. While in the presented case of 2 agents, the separation of the state-space takes the form of a 1-dimensional manifold for any causal policy, in the case of 3 agents, the separation takes the form of 3 separate 2-dimensional manifolds that intersect somewhere in the 3-dimensional state-space. Nevertheless, the paper gives a clear perspective on how to yield such recurrence formulas even for cases of higher dimensions.

Paper 8

AoI in Clocked Networks

Full Paper Title Age-of-Information in Clocked Networks

Publisher In Submission, IEEE/ACM Transactions on Networking

Authors Richard Schöffauer, Gerhard Wunder

Acknowledgment This work is partially funded by the DFG under priority programm SPP 1914 and the BMBF under the 6G-RIC project.

Abstract We derive key features of the Age-of-Information distribution in a system whose activities are strictly limited to periodic instances on a global time grid. In particular, one agent periodically generates updates while the other agent periodically uses the most recently received of those updates. Likewise, transmission of those updates over a network can only occur periodically. All periods may differ. We derive results for two different models: a basic one in which the mathematical problems can be handled directly and an extended model which, among others, can also account for stochastic transmission failure, making the results applicable to instances with wireless communication. For both models, a suitable approximation for the expected Age-of-Information and an upper bound for its largest occurring value are developed. For the extended model (which is the more relevant one from a practical standpoint) we also present numerical results for the distribution of the approximation error for numerous parameter choices. Finally, we show how these results lead to the surprising effect that, in some instances, increasing the network period (slowing down the network) does decrease the expected Age-of-Information.

8.1 Preliminary Remarks

Paper 6 showed that a network policy that minimizes the AoI is very complex as it boils down to a combinatorial problem with non-linear objective function. This raises the question of whether it is possible to predict the AoI in the case of a fixed network policy and some known disturbance patterns for the links. Naturally, one would try and start with the most simple policy, the round-robin policy where communication windows are predetermined and periodic. However, even in this most simple scenario there did not exist a formula with which to determine the expected or peak AoI prior to the following paper. As it turns out, even if communication is assumed to be error-free, the description of the AoI is so complex, that it takes a significant amount of theory to yield the key characteristics of the AoI process, as the paper at hand shows.

8.2 Paper Body

8.2.1 Introduction & Related Research

Timely delivery of data is an important feature of communication systems. To quantify this feature, the Communication-Delay (ComDelay) has long been the predominant metric. In recent years however, especially in the context of machine-to-machine communication (which includes many communication scenarios from IoT), the Age-of-Information (AoI) metric has gained considerable attention. While ComDelay captures the elapsed time between transmission and reception (and thus only considers the communication infrastructure), AoI measures how "old" the current information at the receiver really is. To illustrate the difference, imagine a communication line with a 1 second ComDelay and assume that a packet of information is sent over that line *only* every full hour. The ComDelay is not influenced by this usage and remains 1 second; however the AoI at the receiver is about 30 minutes on average. Every reception, it starts from 1 second and linearly grows with time, peaking at 1 hour and 1 second just before the next reception.

So far, the AoI metric has been studied in a variety of scenarios, an overview over which is given in one of the subsequent paragraphs. However, all these scenarios are concerned with *stochastic* systems: data generation is irregular, intermediate processing takes a random amount of time, transmission is defective. While there certainly is a lot of motivation for such scenarios, it is surprising that the rather simple *deterministic* set-up has been ignored until now. In practice, many autonomous systems do engage in their tasks in a very regular fashion: measuring system-states, processing new outputs, applying corrective actions, they all take place in a predefined amount of time and are repeated thereafter periodically. Hence there is strong motivation to model data generation and utilization as strictly periodic processes that may differ from agent to agent. This is especially true in scenarios where human interaction, and therefore stochastic input, is absent (e.g. industrial production lines). It just so happens that in those scenarios, high requirements on communication (like latency, bandwidth efficiency and robustness against agent drop-outs) do also favor periodic transmissions regimes like token-ring protocols or simple round-robin policies [105].

Therefore, in this paper, we investigate an elementary system that consists of

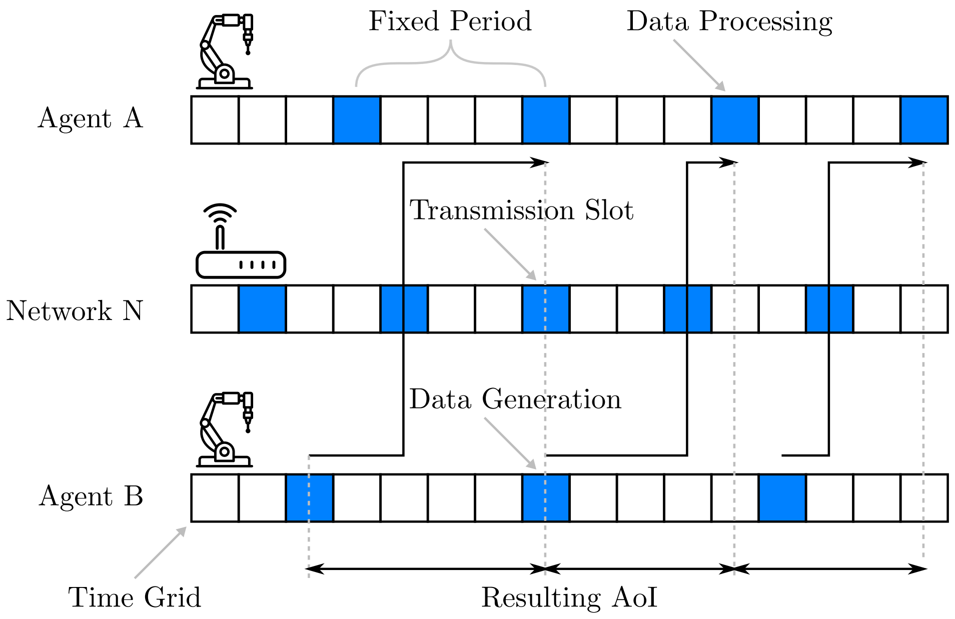


Fig. 8.1: General set-up: two agents communicate over a network. Each agent features its independent working cycle (distance between blue slots) on a shared time grid. Agent B periodically generates status-updates which are send over a network N to agent A. Agent A can only make us of this new information at the start of its next processing cycle.

two subsystems with periodic workflow and a connecting communication network that also features a periodic workflow. It turns out that even without any stochastic effects taken into account, the non-linear behavior of the AoI makes it difficult to yield analytical expressions e.g. for the average AoI or the largest occurring AoI in the system. Our **contribution** consists of deriving expressions for exactly such key performance parameters in connected subsystems with periodic workflow. Along the way, we state structural results regarding the AoI process over time for those systems.

Regarding **related research**, note that to the best of our knowledge, we are the first to investigate the AoI in the context of such a deterministic framework. Therefore the following references are only of limited use when trying to contextualize our contribution. Investigation of the AoI metric started with [106, 107] where the average AoI was optimized over the update generation rate for elementary queueing systems such as M/M/1, M/D/1, and D/M/1. Naturally, this sparked further research exploring various other queueing systems and other AoI features like the peak AoI. The benefit of having multiple servers (e.g. M/M/2 queue) was investigated in [108, 109, 110, 111]. The effect of packet deadlines, after which packets are deleted, freeing up network resources in the process, was demonstrated in [112, 113, 114]. Energy consuming update generation and transmission can be modeled by energy harvesting sources which, in the context of AoI, was investigated in [115, 116, 98, 117, 118]. Direct power constraints on the other hand were considered in [119]. AoI in the context of wireless networks, especially broadcast networks, where link transmission is defective and interference constraints limit the set of admissible scheduling policies was investigated in [96, 89, 120, 121]. Here, the focus lies on finding (near-) optimal policies to minimize the AoI for the entire network. The special case of stochastic policies that trigger transmissions over individual links based on Bernoulli processes (ALOHA-like policies) was researched in

[122, 123]. AoI in multi hop wireless networks was dealt with in [124, 125, 126], and for the special case of gossip networks in [127, 128, 129].

We close the introduction with some notes on the **notation**:

- For sandwiched values we use the following shorthand:

$$z = x \pm y \qquad \Longleftrightarrow \qquad x - y \leq z \leq x + y \tag{8.1}$$

- If not stated otherwise, sets in this paper are multisets, i.e. a single element might have more than one occurrence. (Therefore a multiset might represent a distribution of values.)

- Sets of evenly distributed natural numbers are denoted as the triple ⟨start, step-size, number of steps⟩:

$$\Big\langle y, x, X \Big\rangle := \Big\{ y,\ y + x,\ y + 2x, \dots y + (X-1)x \Big\} \tag{8.2}$$

- The modulo-operator will be abbreviated by %, its complement by $\overline{\%}$:

$$x \,\%\, y := x - \left\lfloor \frac{x}{y} \right\rfloor y \qquad\qquad x \,\overline{\%}\, y := \left\lceil \frac{x}{y} \right\rceil y - x \tag{8.3}$$

(While $x \,\%\, y$ is the distance between x and the closest multiple of y that is still smaller than x, $x \,\overline{\%}\, y$ is the distance between x and the closest multiple of y that is still larger than x. It holds that $(-x) \,\%\, y = x \,\overline{\%}\, y$.)

8.2.2 Basic System Model

Our elementary system consists of two agents (A and B) and a communication network (N). Time is slotted with $t \in \mathbf{N}$ indicating the time-step. Agent A periodically engages in a processing cycle for which he uses the latest status-update he received from agent B. While agent A is processing, he may receive new status-updates from agent B but is not able to make use of them until the start of his next processing cycle. Only the newest status-update from B at A is needed for processing and thus only the last received status-update is buffered. Still, even this status-update is, in general, several time-steps old when it is used at the start of the next processing cycle. This age is called the Age-of-Information, or short AoI. Agent B generates its status-updates periodically. Likewise, the network N can transmit updates from B to A only at periodically distributed time-steps (e.g. because other users require communication resources as well).

In particular: agent A starts a new processing cycle every A-th time-step, agent B generates an update every B-th time-step, network N transmits the latest update from agent B every N-th time-step. These periods A, B, N may share a greatest common divisor (pairwise), labeled a, b, n, and a coprime "core" which remains once the common divisors are removed, labeled A', B', N'. More specifically:

$$\begin{aligned} A &= A' \cdot b \cdot n & & & a &= \gcd(B, N) \\ B &= B' \cdot b \cdot a & &\text{such that} & b &= \gcd(A, B) \\ N &= N' \cdot n \cdot a & & & n &= \gcd(A, N) \end{aligned} \tag{8.4}$$

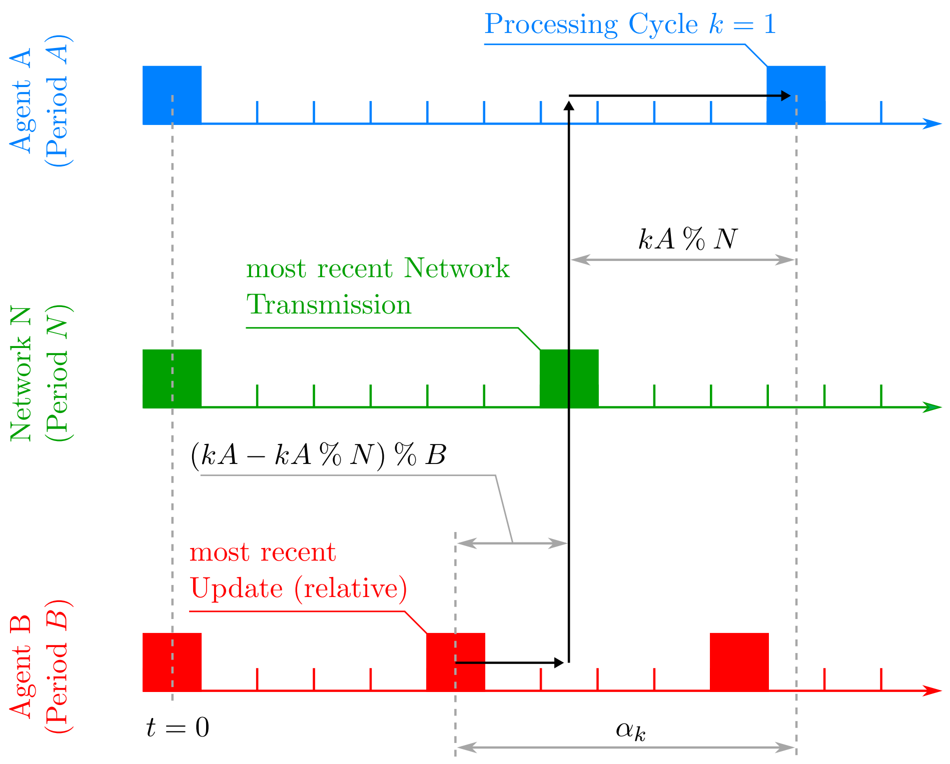


Fig. 8.2: Basic model. All periods start at $t = 0$, data generation and transmission is instantaneous.

All quantities are $\in \mathbb{N}$. We do not consider a divisor (except 1) that divides all three periods A, B, N since such a divisor can readily be accounted for by scaling the entire time axis.

From here, we develop two different models. The first one is labeled the **"basic" model**:

- Update generation and transmission is instantaneous.
- All periods start at $t = 0$.
- Transmissions are always successful.

Figure 8.1 illustrates the setting. Notably, the basic model already exhibits the fundamental mathematical challenges, which is why most of our proofs will address this system-model.

We are interested in the AoI process $(\alpha_k)_{k\in\mathbb{N}}$ (from now on denoted just as (α_k)) where α_k is the AoI of the latest received update (from B at A) at the beginning of agent A's processing cycle k. In case of the basic model, this process can readily be deduced from Figure 8.1 as

$$\begin{aligned}\alpha_k &= kA \,\%\, N + (kA - kA \,\%\, N) \,\%\, B \\ &= kA \,\%\, N + (kA \,\%\, B - kA \,\%\, N) \,\%\, B\end{aligned} \tag{8.5}$$

Even though (α_k) is deterministic, we will interpret (α_k) as a stochastic process since its behavior can quickly become quite complex.

As a direct consequence of this kind of modeling, in time-step $t = 0$, agent B generates a status-update which network N immediately transmits to agent A and which can then be used in agent A's first processing cycle. Hence, in time-step $t = 0$ (and periodically thereafter) the AoI is zero: $\alpha_0 = 0$, which already motivates a second system-model.

8.2.3 Extended System Model

Intuitively, the AoI at agent A should never truly be zero for several reasons, e.g. due to the time it takes for the transmission to reach agent A or due to the time it takes agent B to generate the status-update. Therefore, we will extend our results to a system-model which accounts for such delays and also introduces some other aspects that are quite common in this field of research. We refer to this new model as the **"extended" model**:

- Update generation and transmission each take one time-step.
- Relative to time-step $t = 0$, the first periods of agent B and network N are delayed by Δ_B and Δ_N time-steps.
- Transmissions succeed only with probability $p = 1 - \bar{p}$.

For a visual representation see Figure 8.3.

In this model we denote the AoI process with (ε_k) while its (verbal) definition stays the same relative to the model. Compared to the basic model, the elapsed time since the last network transmission relative to the current time-step $t = kA$ (the beginning of agent A's k-th processing cycle) changes according to

$$\begin{aligned} & kA \,\%\, N && (8.6)\\ \hookrightarrow\quad & (kA - \Delta_N) \,\%\, N && (8.7)\\ \hookrightarrow\quad & 1 + (kA - \Delta_N - 1) \,\%\, N && (8.8)\\ \hookrightarrow\quad & lN + 1 + (kA - \Delta_N - 1) \,\%\, N && (8.9) \end{aligned}$$

The first change follows from the shift of network N's period. The second change follows from the transmission delay: (8.7) and (8.8) are identical if the value of (8.7) is from the set $\{1, \dots N-1\}$. For these values the transmission delay does not effect the outcome since there is enough time until agent A's processing cycle starts. However, if transmission takes place in the same time-step as A's processing cycle starts (eq. (8.7) yields 0), then *in the extended* model, agent A cannot make use of the new update yet and has to resort to the previous transmission which is N steps in the past (eq. (8.8) yields N). The third change follows from unsuccessful transmissions. If the latest l transmissions did not succeed, (8.9) accounts for that by adding lN time-steps on top of (8.8). We will account for the probability of that happening later.

On agent B's side, the elapsed time since the last status-update from agent B that the network could use for transmission becomes

$$\begin{aligned} & (kA - \bigstar) \,\%\, B && (8.10)\\ \hookrightarrow\quad & (kA - \bigstar - \Delta_B) \,\%\, B && (8.11)\\ \hookrightarrow\quad & 1 + (kA - \bigstar - \Delta_B - 1) \,\%\, B && (8.12) \end{aligned}$$

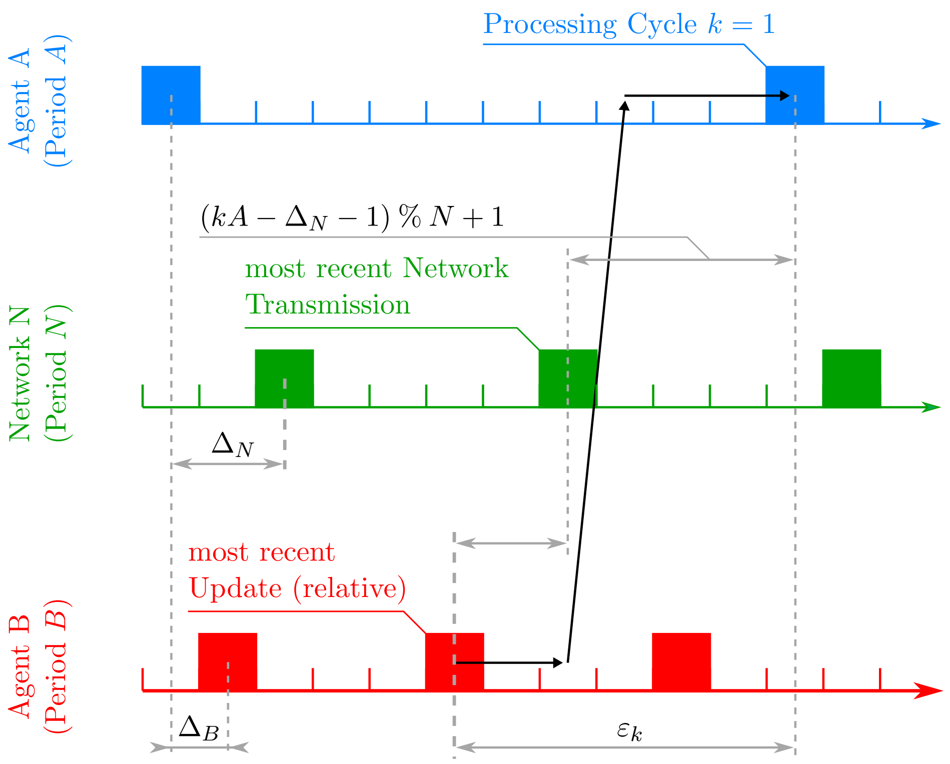


Fig. 8.3: Extended model. Start of initial periods are shifted, update generation and transmission each take up one time-step.

where ★ is the placeholder for the term (8.9), i.e. the elapsed time since the latest network transmission (see. Figure 8.3). As before, the changes are due to the shift of B's period and the update generation delay.

Simply summing (8.9) and (8.12) seems to yields an evolution for (ε_k). However, this evolution does not account for the fact that the number l of failed transmissions prior to reception most certainly changes over time. Hence, such a sum only yields an evolution under the assumption that for every kA-th time-step, the last l transmissions did fail while the $(l+1)$-th transmission did succeed. We denote the sequence following such an evolution with $(\varepsilon_k^{[l]})$ (ignoring the fact that it might be impossible to comply with sich an assumption):

$$\begin{aligned}\varepsilon_k^{[l]} &= 2 + lN + (kA - \Delta_N - 1)\,\%\,N \\ &\quad + \Big[(kA - \Delta_B - 2 - lN)\,\%\,B - (kA - \Delta_N - 1)\,\%\,N\Big]\,\%\,B\end{aligned} \tag{8.13}$$

Though this looks quite more intricate than (8.5), we can analyze $(\varepsilon_k^{[l]})$ the same way we analyzed the basic model. Note that the values 2 and 1, which appear in (8.13), stem from the delays in update generation and transmission. It is an easy exercise to use different delay values and carry those over to any of the presented results.

8.2.4 Results for the Basic Model

Treating (α_k) as a stochastic process, the main goal of this paper is to find usable expressions for the expectation and some upper bound on its largest values. As it turns out, these expressions can be derived, once the following structural result on the distribution of values of (α_k) is established:

Theorem 6. *In the basic model, the period of the process (α_k) is $aB'N'$. And over $aB'N'$ consecutive elements, the AoI sequence (α_k) generates the following distribution of values:*

$$\bigcup_{k=1}^{aB'N'} \left\{\alpha_k\right\} = \bigcup_{\substack{i=0\dots a-1\\ j=0\dots N'-1}} \left\langle c_{ij}, ab, B' \right\rangle \tag{8.14}$$

with

$$c_{ij} = iA\,\%\,(ab) + \left\lceil \frac{iA\,\%\,(an) - iA\,\%\,(ab) + jan}{ab} \right\rceil ab \tag{8.15}$$

Theorem 6 reveals that the distribution of (α_k) can be expressed as a superposition of sets of equally distanced values. This is illustrated in Figure 8.4. The most influential quantities in this regard are the starting positions of these sets: c_{ij}. The ceiling and modulo operators in (8.15) lead to an intricate expression for the expected AoI which does not lend itself to practical use (derivation in the appendix):

$$\begin{aligned}\mathbb{E}[\alpha_k] = {} & \frac{B+N-n+ab}{2} - \frac{b}{N'}\Bigg(\frac{a(b+1)}{2}\left\lfloor\frac{N'}{b}\right\rfloor \\ & + \left\lceil (N'\,\%\,b)\frac{a}{b}\right\rceil + \sum_{i=0}^{a-1}\sum_{j=0}^{N'\%b-1} \frac{\left(j - \lfloor\frac{ib}{a}\rfloor\right) n\,\%\,b}{b}\Bigg)\end{aligned} \tag{8.16}$$

However, it can be simplified considerably by using the central property of the ceiling operator: $x \leq \lceil x \rceil < x+1$, which yields the following Corollary:

Corollary 1. *In the basic model, the expected AoI is bounded by*

$$\mathbb{E}[\alpha_k] = \frac{B+N-n}{2} \pm \frac{ab}{2} \tag{8.17}$$

Using $\hat{\mathbb{E}}[\alpha_k] = \frac{B+N-n}{2}$ as an approximation, the maximal relative error becomes

$$\left| \frac{\mathbb{E}[\alpha_t] - \hat{\mathbb{E}}[\alpha_t]}{\mathbb{E}[\alpha_t]} \right| \leq \frac{ab}{(B'-1)ab + n(aN'-1)} \tag{8.18}$$

The largest value of (α_k) is

$$\max_k \alpha_k \leq B + N - n \tag{8.19}$$

Proofs for both Theorem 6 and Corollary 1 are found in the appendix. Note that according to Corollary 1, it is possible to increase A' (the part of A that is coprime to all other quantities) without changing any of the results. Furthermore, when using the center of the possible range ($\hat{\mathbb{E}}[\alpha_k]$) as an approximation for the true value $\mathbb{E}[\alpha_k]$, the relative error can become infinitely large if $B' = 1 = aN'$. The reason lies in the fact that (α_k) becomes a null-sequence for those parameters. Since the basic model rather serves theoretical purposes, we omit any further investigation in the quality of the approximation.

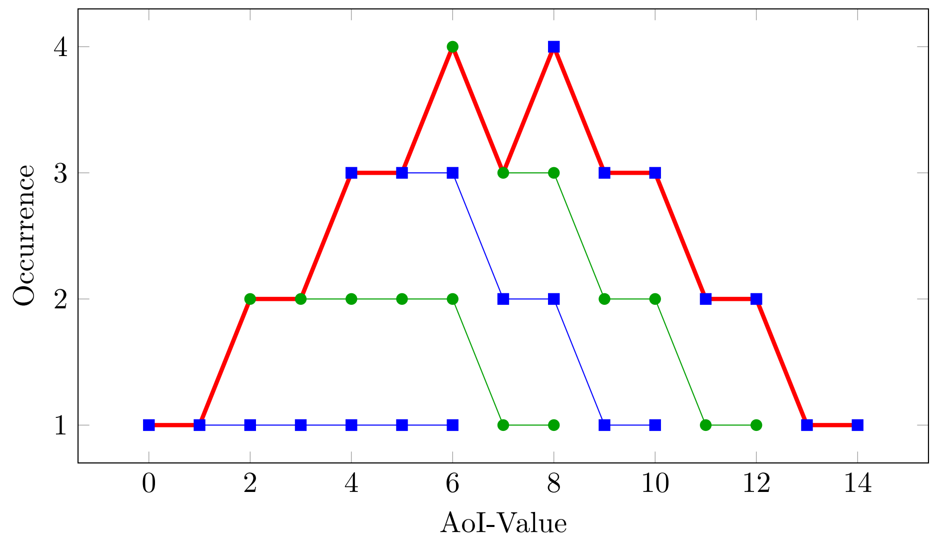


Fig. 8.4: Visualization of the structure implied by Theorem 6: 5 sets of equally distanced values (connected marks), each starting with a different value. In superposition they yield the desired distribution (red). Parameters: $A = 17, B = 7, N = 5, a = b = 1, n = 2$.

8.2.5 Results for the Extended Model

In analogy to the results for the basic model, we can find the same structural result for the distribution of the AoI under the extended model. Note however, that Theorem 7 is restricted to a sequence $(\varepsilon_k^{[l]})$, i.e. a specific value of l. The parameter l stands for the assumption that, no matter which processing cycle of agent A is considered, the last $l-1$ network transmissions prior to that cycle did fail. Though this might be an assumption that is impossible to fulfill, the theorem will still be crucial for following results.

Theorem 7. *In the extended model, the period of the process* $(\varepsilon_k^{[l]})$ *is* $aB'N'$*. And over* $aB'N'$ *consecutive elements, the AoI sequence* $(\varepsilon_k^{[l]})$ *generates the following distribution of values:*

$$\bigcup_{k=1}^{aB'N'} \left\{\varepsilon_k^{[l]}\right\} = \bigcup_{\substack{i=0\ldots a-1\\ j=0\ldots N'-1}} \left\langle c_{ij}^{[l]}, ab, B' \right\rangle \tag{8.20}$$

with

$$c_{ij}^{[l]} = 2 + lN + (iA - \Delta_B - 2 - lN) \,\%\, ab + \left\lceil \frac{(iA - \Delta_N - 1) \,\%\, an - (iA - \Delta_B - 2 - lN) \,\%\, ab + jan}{ab} \right\rceil ab \tag{8.21}$$

The similarities between Theorem 6 and Theorem 7 are obvious. Notably, the change in the structure is restricted to the starting points c_{ij} of the sets with equally distanced elements. As before, an exact equation for $\mathbb{E}[\varepsilon_k]$ can be developed (combining techniques from the proofs of (8.16) and Corollary 2):

$$\mathbb{E}[\varepsilon_k] = \frac{B+N-ab-n}{2} + K + \frac{\bar{p}}{p}N \tag{8.22}$$

$$+ \lim_{L\to\infty} \sum_{l=0}^{L} \frac{p\bar{p}^l}{aN'} \sum_{\substack{i=0,\dots a-1\\ j=0,\dots N'-1}} \left(\Delta_B - \Delta_N + lN + 1 + jan - \left\lfloor \frac{iA-\Delta_N-1}{an} \right\rfloor an\right) \overline{\%}\, ab$$

The following corollary, crucially, is also not restricted on the impossible assumption on $(\varepsilon_k^{[l]})$. The exact reason for this is provided in the proof and is based on the probabilistic occurrence of the values of $(\varepsilon_k^{[l]})$ in (ε_k).

Corollary 2. *In the extended model, the expected AoI is bounded by*

$$\mathbb{E}[\varepsilon_k] = \frac{B+N-n}{2} + K + \frac{\bar{p}}{p}N \pm \frac{ab}{2} \tag{8.23}$$

with

$$K = 2 + (\Delta_N + 1)\, \overline{\%}\, n \tag{8.24}$$

Using $\hat{\mathbb{E}}[\varepsilon_k] = \frac{B+N-n}{2} + K + \frac{\bar{p}}{p}N$ *as approximation, the maximal relative error becomes*

$$\left| \frac{\mathbb{E}[\varepsilon_k] - \hat{\mathbb{E}}[\varepsilon_k]}{\mathbb{E}[\varepsilon_k]} \right| \leq \frac{ab}{(B'-1)ab + n(aN'-1) + 2(K + \frac{\bar{p}}{p}N)} \tag{8.25}$$

With probability σ *it holds* $\forall\, k$ *that*

$$\varepsilon_k \leq B + N - n + K + N \left\lceil \frac{\ln(1-\sigma)}{\ln(1-p)} - 1 \right\rceil \tag{8.26}$$

8.2.6 Approximation Error

The structure of (8.22) makes it very hard to analyze the relative approximation error (LHS of (8.26)) analytically. Therefore, we utilize simulations to validate its suitability. Note however, that the use of a Monte-Carlo protocol is a rather poor choice here: It is easy to check that for 3 random variables $X, Y, Z \in \{1, 2, \dots M\} \subset \mathbb{N}$ we have

$$\mathbb{E}\left[\frac{\gcd(X,Y)\cdot\gcd(Y,Z)}{X}\right] \to 0 \tag{8.27}$$

as M grows to infinity [130] (essentially because $\gcd(X,Y)$ grows logarithmically). Parameter M is our simulation range and is needed since one cannot generate arbitrarily large random numbers. Hence, if M is chosen large enough, a numerical evaluation of the LHS of (8.26) via randomly chosen parameter configurations will result in a negligible error.

However, ever larger values for M would mean ever larger values for A, B, N and thus would mean that the time grid becomes ever finer, which is unsuitable for practical purposes. Therefore, it is prudent to analyze the LHS of (8.26) for rather small values of M. Consequently, a Monte-Carlo protocol (with large M) can be omitted in favor of a full blown simulation over the entire parameter space (bounded by a small M).

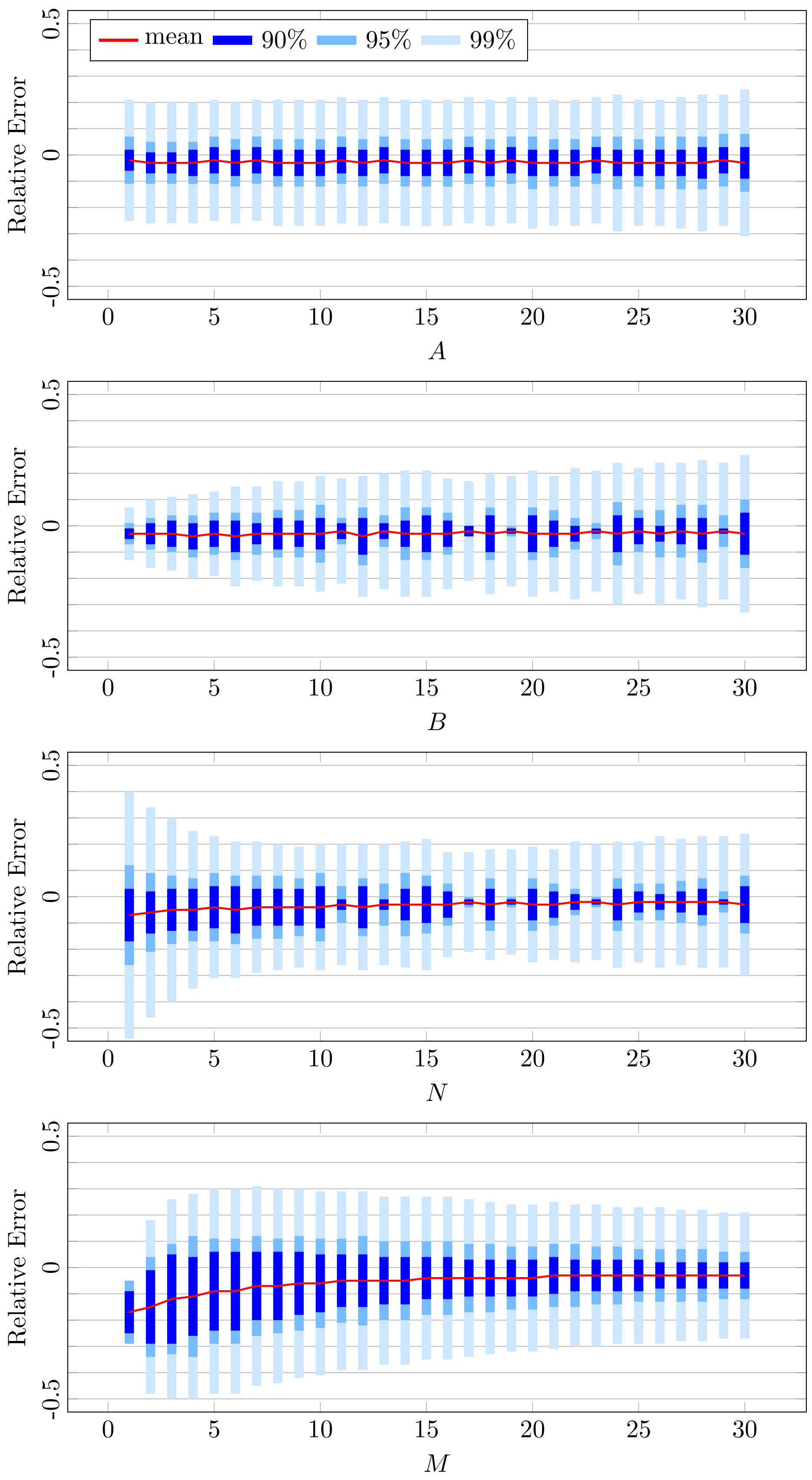


Fig. 8.5: Distribution of the relative approximation error (8.26) as function of the agents periods A, B, N and simulation range M (sign sensitive). Colored bars indicate intervals in which an overwhelming amount of data points lie in.

In the following, we evaluated all relative errors for $M \leq 30$ such that

$$\begin{gathered} 1 \leq A, B, N \leq 30 \\ 0 \leq \Delta_B < B \\ 0 \leq \Delta_N < N \\ p \in \{0.1,\ 0.2,\ \ldots,\ 0.9,\ 1\} \end{gathered} \tag{8.28}$$

resulting in just under $65 \cdot 10^6$ data points. Each resulting distribution (a function of the respective parameter) is visualized by its mean and 3 encompassing intervals, indicating where a certain majority (99%, 95%, 90%) of its values can be found.

The most relevant result is illustrated in the last plot of Figure 8.5, showing that the mean error and its variance tend to shrink as M grows. Specifically, for $M = 30$, the mean relative error is roughly -3% and 95% of all evaluated errors are between -12% and 6%. The negative sign for the mean indicates that our approximation does yield a conservative result, overestimating the average AoI.

8.2.7 Implication on Network Parametrization

The average AoI, $\mathbb{E}[\varepsilon]$, is roughly governed by the term $B + N - \gcd(A, N)$. While B and N do increase $\mathbb{E}[\varepsilon]$, the greatest common divisor $n = \gcd(A, N)$, shared by the network's period N and agent A's period A, decreases it. Hence, a smaller (faster) N may not yield a smaller (faster) $\mathbb{E}[\varepsilon]$.

Taking the viewpoint of the network, we can treat A as a random quantity (as there are several scenarios in which the agents' periods are unknown to the network). Doing so, we can capture this effect by finding an expression for the expected greatest common divisor $\mathbb{E}[\gcd(R, N)]$ where R is a substitution for A and stands for a random integer. In case that R is drawn uniformly from all $\mathbb{N}_+$, we state the following theorem:

Theorem 8. *Let $R \in \mathbb{N}_+$ be a random positive integer whose distribution is uniform over $\mathbb{N}_+$. Further let $M = \prod_{x \in \mathcal{X}} x^{r(x)}$ be an integer M together with its prime factorization. Then the expected greatest common divisor shared by M and R is*

$$\mathbb{E}[\gcd(R, M)] = \sum_{\mathcal{X}' \subset \mathcal{X}} \prod_{x \in \mathcal{X}'} r(x) \sum_{\mathcal{X}'' \subset \mathcal{X}'} (-1)^{|\mathcal{X}''|} \prod_{x \in \mathcal{X}''} \frac{1}{x} \tag{8.29}$$

In Figure 8.6, this result is used to plot $\mathbb{E}[\gcd(R, N)]$ over N. It can be seen that e.g. $N = 60$ is a more beneficial network period than $N = 59$, as the choice $N = 60$ will, on average, yield a smaller AoI. This is a direct consequence of the number 59 being a prime and the number 60 having much more divisors than any number in its immediate neighborhood. (A large number of divisors increases the chance of n being large.) Given that the greatest common divisor only grows logarithmically, this effect becomes less important the larger the network period N gets, as the linear growth of N will dominate the average AoI.

8.2.8 Conclusion

For the described system (see Figure 8.3), we derived a structural result on the distribution of the values of the AoI process. Based on that result we developed both

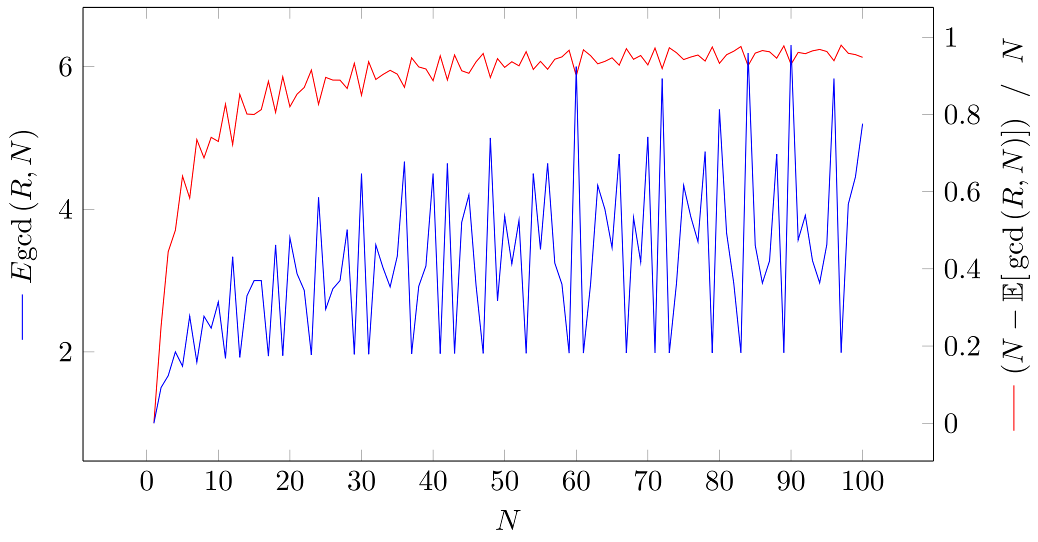


Fig. 8.6: Expected greatest common divisor of a random positive integer R (uniformly distributed over its entire domain) and another integer N.

an exact and an approximating expression for the mean of the process together with a probabilistic bound on the maximal occurring AoI. The simulations show that the approximation is good enough to enable efficient calculation of the mean AoI with limited error. We also obtained the surprising result that, in some instances, it can be beneficial to slow the network down in order to decrease the AoI.

8.2.9 Appendix

To present the proofs as succinct as possible, we will often use the notation $\mathcal{Y} = \mathcal{X} + x$ to express that the set $\mathcal{Y}$ is generated by adding the value x to every element of the set $\mathcal{X}$.

Lemma 5. *Let $X, Y \in \mathbb{N}$ and $\gcd(X,Y) = 1$ (i.e. X and Y are coprime). Then*

$$\bigcup_{k=1}^{Y} \left\{ kX \,\%\, Y \right\} = \left\langle 0, 1, Y \right\rangle \tag{8.30}$$

Proof. We prove via contradiction. Denote the following residues, starting with arbitrary $k \in \mathbb{N}$ as

$$\begin{aligned} kX \,\%\, Y &= z_0 \\ (k+1)X \,\%\, Y &= z_1 \\ &\vdots \\ (k+Y-1)X \,\%\, Y &= z_{Y-1} \\ (k+Y)X \,\%\, Y &= kX \,\%\, Y = z_0 \end{aligned} \tag{8.31}$$

with $z_i \in \{0, 1, \dots Y-1\}$. Suppose that there are at least two numbers from $\{z_0, z_1, \dots z_{Y-1}\}$ which are equal, say $z_{k'}$ and $z_{k''}$, then

$$\begin{aligned} (k+k')X \,\%\, Y &= (k+k'')X \,\%\, Y \\ \Rightarrow (k+k')X - (k+k'')X &= (k'-k'')X = wY \end{aligned} \tag{8.32}$$

for some $w \in \mathbb{N}$. Per construction we have $|k' - k''| < Y$ which implies that $\mathrm{lcm}(X, Y) = |k' - k''|X < YX$. This must be wrong since X and Y are coprime, proving the claim. □

Lemma 6. *Let (x_k) be a sequence with period X and denote its values according to $(x_{1+lX}, \dots x_{X+lX}) = (\mathfrak{x}_1, \dots \mathfrak{x}_X)$ for all $l \in \mathbb{N}$. Likewise let (y_k) be a sequence with period Y and $(y_{1+lY}, \dots y_{Y+lY}) = (\mathfrak{y}_1, \dots \mathfrak{y}_Y)$. Further let $z = \gcd(X, Y)$. Then over any $\frac{XY}{z}$ consecutive steps k, the set of generated pairs (x_k, y_k) contains exactly each pairing $(\mathfrak{x}_i, \mathfrak{y}_j)$ once, whose indices are part of the same residual class r with regard to z:*

$$\bigcup_{k=1}^{\frac{XY}{z}} \Big\{(x_k, y_k)\Big\} = \bigcup_{r=1}^{z} \bigcup_{\substack{i \in \left\langle r, z, \frac{X}{z} \right\rangle \\ j \in \left\langle r, z, \frac{Y}{z} \right\rangle}} \Big\{(\mathfrak{x}_i, \mathfrak{y}_j)\Big\} \tag{8.33}$$

Proof. For $z = 1$, (8.33) means that every possible pairing occurs exactly once. Conversely, assume that a specific pairing $(\mathfrak{x}_i, \mathfrak{y}_j)$ is generated twice during the XY steps and denote these steps with k' and k''. Then $|k'' - k'| = \Delta < XY$ and, due to the periodicity, both X and Y must be divisors of Δ, making Δ a common multiple smaller XY. Since X and Y are coprime this is a contradiction.

For $z \in \mathbb{N}$, construct two new sequences $(\tilde{x}_\kappa)$ and $(\tilde{y}_\kappa)$ according to

$$\begin{aligned} \tilde{x}_\kappa &= (x_{\kappa z+1}, \dots x_{\kappa z+z}) \\ \tilde{y}_\kappa &= (y_{\kappa z+1}, \dots y_{\kappa z+z}) \end{aligned} \tag{8.34}$$

I.e. one element of $(\tilde{x}_\kappa)$ consists of z consecutive elements from (x_k). Per construction $(\tilde{x}_\kappa)$ has periodicity $\frac{X}{z}$ and $(\tilde{y}_\kappa)$ has periodicity $\frac{Y}{z}$. Also, $\frac{X}{z}$ and $\frac{Y}{z}$ are coprime. Hence, according to the first part of the proof, over any $\frac{X}{z} \cdot \frac{Y}{z}$ consecutive steps, each of the $\frac{X}{z}$ different elements of $(\tilde{x}_\kappa)$ will eventually pair with each of the $\frac{Y}{z}$ different elements of $(\tilde{y}_\kappa)$. Using (8.34) this implies that over any $\frac{XY}{z^2} \cdot z$ consecutive steps of the original sequence, each $x_{iz+\kappa}$ meets every $y_{jz+\kappa}$ exactly once, with $i = 1, \dots \frac{X}{z}$; $j = 1, \dots \frac{Y}{z}$; $\kappa = 1 \dots z$. With slight redefinition of i and j (such that they include z) this means that for given κ, $\mathfrak{x}_{i+\kappa}$ meats $\mathfrak{y}_{j+\kappa}$ iff $i \in \{0, z, 2z, \dots \left(\frac{X}{z} - 1\right) z\}$ and $j \in \{0, z, 2z, \dots \left(\frac{Y}{z} - 1\right) z\}$. The proposition follows. □

Lemma 7. *Let $X, x, y \in \mathbb{N}$, then*

$$\Big\langle y, x, X \Big\rangle \,\%\, Xx = \Big\langle y \,\%\, x, x, X \Big\rangle \tag{8.35}$$

$$\Big\langle -y, x, X \Big\rangle \,\%\, Xx = \Big\langle y \,\overline{\%}\, x, x, X \Big\rangle \tag{8.36}$$

Proof. We proof (8.36) and initially assume that $y < Xx$. Then $\langle -y, x, X \rangle$ will consist of positive and possibly negative values. Continuously adding x to $-y$ eventually starts producing non-negative results, beginning with the $\lceil \frac{y}{x} \rceil$-th summation step. This motivates the separation

$$\Big\langle -y, x, X \Big\rangle = \underbrace{\Big\langle -y, x, \Big\lfloor \frac{y}{x} \Big\rfloor \Big\rangle}_{\text{negative values}} \cup \underbrace{\Big\langle -y + \Big\lceil \frac{y}{x} \Big\rceil x, x, X - \Big\lfloor \frac{y}{x} \Big\rfloor \Big\rangle}_{\text{non-negative values}} \tag{8.37}$$

The largest term of the "non-negative" set is $-y + (X-1)x$. Therefore, taking all values modulo (Xx) only effects the "negative" set by adding Xx. Doing so, the smallest term in the "negative" set becomes $-y + Xx$ (which is exactly one x greater than the largest term of the "non-negative" set). Hence, after the $\% \,(Xx)$ operation, the elements of both sets can be rearranged into a new set according to

$$\begin{aligned}\left\langle -y, x, X\right\rangle \% \,(Xx) &= \left\langle -y + Xx, x, \left\lfloor \frac{y}{x} \right\rfloor \right\rangle \cup \left\langle -y + \left\lceil \frac{y}{x} \right\rceil x, x, X - \left\lfloor \frac{y}{x} \right\rfloor \right\rangle \\ &= \left\langle -y + \left\lceil \frac{y}{x} \right\rceil x, x, X \right\rangle \end{aligned} \tag{8.38}$$

Suppose now that $y \geq Xx$ and in particular $y \,\%\, Xx = z$. Then we get

$$\begin{aligned}\left\langle -y, x, X\right\rangle \% \,Xx &= \left\langle -y \,\%\, Xx, x, X\right\rangle \% \,Xx \\ &= \left\langle -z, x, X\right\rangle \% \,Xx \\ &\overset{*}{=} \left\langle -z + \left\lceil \frac{z}{x} \right\rceil x, x, X\right\rangle \\ &\overset{**}{=} \left\langle -y + \left\lceil \frac{y}{x} \right\rceil x, x, X\right\rangle \end{aligned} \tag{8.39}$$

where (*) is true because $z < Xx$ and (**) is readily verified by substitution ($y = wXx + z$ for some $w \in \mathbb{N}$). To proof (8.35), see that

$$(-y) \,\%\, x = (-y) - \left\lfloor \frac{(-y)}{x} \right\rfloor x = -y + \left\lceil \frac{y}{x} \right\rceil x = y \,\overline{\%}\, x \tag{8.40}$$

and hence substitution of $-y$ for y in (8.36) yields (8.35). □

Proof of Theorem 1

The evolution of the AoI sequence (α_k) according to (8.5) can be interpreted as a function of the sequences (η_k) and (β_k), identified by:

$$\begin{aligned}\alpha_k = \alpha_k(\beta_k, \eta_k) = \eta_k + (\beta_k - \eta_k) \,\%\, B = \\ \underbrace{kA \,\%\, N}_{=:\, \eta_k} + \big(\underbrace{kA \,\%\, B}_{=:\, \beta_k} - \underbrace{kA \,\%\, N}_{=:\, \eta_k}\big) \,\%\, B\end{aligned} \tag{8.41}$$

This means that

$$\begin{aligned}\beta_k &= b\big[k(nA') \,\%\, (aB')\big] \\ \eta_k &= n\big[k(bA') \,\%\, (aN')\big]\end{aligned} \tag{8.42}$$

and we know from Lemma 5 that (β_k) and (η_k) do generate the sets $\langle 0, b, aB'\rangle$ and $\langle 0, n, aN'\rangle$ (over their respective periods). With one element from each set, we can readily describe the resulting value for α_k. However, not all possible element pairings are realized. Our first priority is therefore to determine which elements (and values) are realized simultaneously. To that end it is prudent to denote the elements in appearing order and assign to them "fixed" values $\mathfrak{b}_i$ and $\mathfrak{n}_i$ according to

$$\begin{aligned} \beta_{1+l(aB)} &=: \mathfrak{b}_1 & \eta_{1+l(aN)} &=: \mathfrak{n}_1 \\ \beta_{2+l(aB)} &=: \mathfrak{b}_2 & \eta_{2+l(aN)} &=: \mathfrak{n}_2 \\ &\vdots & &\vdots \\ \beta_{aB+l(aB)} &=: \mathfrak{b}_{aB} & \eta_{aN+l(aN)} &=: \mathfrak{n}_{aN} \end{aligned} \tag{8.43}$$

with $l \in \mathbb{N}$.

Since the periods of (β_k) and (η_k) are aB' and aN', the period of (α_k) must be $aB'N'$. From Lemma 6, we know that during those $aB'N'$ steps only those $\mathfrak{b}_i$ and $\mathfrak{n}_j$ will occur at the same step, whose indices belong to the same residue class $r \in \mathbb{N}$ with respect to the greatest common divisor a:

$$\bigcup_{k=1}^{aB'N'} \Big\{\alpha_k(\beta_k, \eta_k)\Big\} = \bigcup_{r=0}^{a-1} \bigcup_{i=1}^{B'} \bigcup_{j=1}^{N'} \{\alpha_k(\mathfrak{b}_{ia+r}, \mathfrak{n}_{ja+r})\} \tag{8.44}$$

Looking at the value behind $\mathfrak{b}_{ia+r}$ we have

$$\begin{aligned} \mathfrak{b}_{ia+r} &= (ia+r)A \,\%\, B \\ &= (iaA \,\%\, B + rA) \,\%\, B \\ &= (iabnA' \,\%\, abB' + rA) \,\%\, B \\ &= (ab(inA' \,\%\, B') + rA) \,\%\, B \end{aligned} \tag{8.45}$$

Over all possible i, i.e. for $i = 1, \dots B'$, the term $ab(inA' \,\%\, B')$ generates the set $\big\langle 0, ab, B' \big\rangle$. And therefore, using Lemma 7

$$\bigcup_{i=1}^{B'} \Big\{\mathfrak{b}_{ia+r}\Big\} = \left(\left\langle 0, ab, B' \right\rangle + rA\right) \,\%\, abB = \left\langle rA \,\%\, ab, ab, B' \right\rangle \tag{8.46}$$

Of course, the same holds true for $\mathfrak{n}_{ja+r}$:

$$\bigcup_{j=0}^{N'} \Big\{\mathfrak{n}_{ja+r}\Big\} = \left\langle rA \,\%\, an, an, N' \right\rangle \tag{8.47}$$

Knowing that, we can look back at our starting equation:

$$\alpha_k = \eta_k + (\beta_k - \eta_k) \,\%\, B \tag{8.48}$$

Over (α_k)'s period $aB'N'$, every value from (8.46) will pair up with every value from (8.47) exactly once. I.e. for the value $\mathfrak{n}_{ja+r}$ from (8.47) the sequence (α_k) generates the values

$$\mathfrak{n}_{ja+r} + \left(\left\langle rA \,\%\, ab, ab, B' \right\rangle - \mathfrak{n}_{ja+r}\right) \,\%\, B \tag{8.49}$$

which according to Lemma 7 is equal to

$$\left\langle rA \,\%\, ab + \left\lceil \frac{\mathfrak{n}_{ja+r} - rA \,\%\, ab}{ab} \right\rceil ab, ab, B' \right\rangle \tag{8.50}$$

To yield an expression for all generated values, we have to take the union of (8.50) over all possible $\mathfrak{n}_{ja+r}$. I.e. the union over all possible residue classes $r = 0, \dots a-1$, and per residue class over all $j = 0, \dots N'-1$. Since we do not care for the order of the generated values, we can invoke (8.47) for the union over j and obtain

$$\bigcup_{j=0}^{N'-1} \left\langle c_{rj}, ab, B' \right\rangle$$
$$\text{with} \qquad c_{rj} = rA \,\%\, ab + \left\lceil \frac{rA \,\%\, an + jan - rA \,\%\, ab}{ab} \right\rceil ab \tag{8.51}$$

Replacing r with i and taking the union over $i = 0, \dots a-1$ yields the proposition. $\square$

Derivation of Equation (8.16)
To obtain the expected AoI, we need an expression for the expected value of c_{ij}, because

$$\begin{aligned}\mathbb{E}[\alpha_k] = \frac{1}{aB'N'} \sum_{k=1}^{aB'N'} \alpha_k &= \frac{1}{aB'N'} \sum_{\substack{i=0\dots a-1\\ j=0\dots N'-1}} \sum_{x \in \left\langle c_{ij}, ab, B' \right\rangle} x \\ &= \frac{1}{aN'} \sum_{\substack{i=0\dots a-1\\ j=0\dots N'-1}} c_{ij} + \frac{(B'-1)ab}{2}\end{aligned} \tag{8.52}$$

Looking at c_{ij} as defined in (8.15), the term containing the ceiling operator can be developed in the following way if we consider its sum over $i = 0, \dots a-1$ *and* $j = 0, \dots N'-1$:

$$\begin{aligned} &\sum \left\lceil \frac{iA \,\%\, an + jan - iA \,\%\, ab}{ab} \right\rceil \\ \overset{(1)}{=} &\sum \left\lceil \frac{n\left(ib - \left\lfloor \frac{ib}{a} \right\rfloor a\right) + jan - b\left(in - \left\lfloor \frac{in}{a} \right\rfloor a\right)}{ab} \right\rceil \\ = &\sum \left\lfloor \frac{in}{a} \right\rfloor + \sum \left\lceil \frac{\left(j - \left\lfloor \frac{ib}{a} \right\rfloor\right) n}{b} \right\rceil \end{aligned} \tag{8.53}$$

Equality (1) holds because of (8.4) and (8.3) and the fact that the factor A' merely reorders the occurrence of the rest classes and can therefore be omitted due to the sum (see Lemma 5).

Applying the substitution

$$\frac{x}{y} = \underbrace{\frac{x - x \,\%\, y}{y}}_{\in \mathbb{N}} + \frac{x \,\%\, y}{y} \tag{8.54}$$

to the remaining ceiling and floor operators makes it possible to retract the integer part from the operators. The resulting sums are readily evaluated except the following two:

$$\sum \left\lceil \frac{\left(j - \left\lfloor \frac{ib}{a} \right\rfloor\right) n \,\%\, b}{b} \right\rceil = aN - a \left\lfloor \frac{N'}{b} \right\rfloor - \left\lceil (N' \,\%\, b) \frac{a}{b} \right\rceil \tag{8.55}$$

and

$$\sum \frac{\left(j - \lfloor \frac{ib}{a} \rfloor\right) n \,\%\, b}{b} = \frac{a(b-1)}{2} \left\lfloor \frac{N'}{b} \right\rfloor + \sum_{i=0}^{a-1} \sum_{j=0}^{N'\%b-1} \frac{\left(j - \lfloor \frac{ib}{a} \rfloor\right) n \,\%\, b}{b} \tag{8.56}$$

Equation (8.55) follows since the ceiling operator always evaluates to 1 except when $j - \lfloor \frac{ib}{a} \rfloor$ is a multiple of b. In those cases it evaluates to 0. For $i = 0$ this happens $\lfloor \frac{N'}{b} \rfloor + 1$ times. This does not change as long as $\lfloor \frac{ib}{a} \rfloor < (N'-1) \,\%\, b$, i.e. $\lceil (N' \,\%\, b) \frac{a}{b} \rceil$ times. For all other cases a multiple of b is only realized $\lfloor \frac{N'}{b} \rfloor$ times.

Equation (8.56) follows because any b consecutive summands over j evaluate to $\frac{b-1}{2}$. And there are only $\lfloor \frac{N'}{b} \rfloor$ such sequences of summands over j.

Rejoining all of these results, one can yields the following expression for the sum over all c_{ij}:

$$\begin{aligned} \frac{1}{aN'} \sum_{\substack{i=0,\dots a-1 \\ j=0,\dots N'-1}} c_{ij} = \frac{N-n+2ab}{2} - \frac{b}{N'} \Bigg(\frac{a(b+1)}{2} \left\lfloor \frac{N'}{b} \right\rfloor \\ + \left\lceil (N' \,\%\, b) \frac{a}{b} \right\rceil + \frac{1}{b} \sum_{i=0}^{a-1} \sum_{j=0}^{N'\%b-1} \left(j - \left\lfloor \frac{ib}{a} \right\rfloor \right) n \,\%\, b \Bigg) \end{aligned} \tag{8.57}$$

Finally (8.16) results by substituting (8.57) into (8.52). □

Proof of Corollary 1

To obtain the approximation (8.17), we take (8.15) at face value and use the bounds of the ceiling operator. This gives

$$c_{ij} = iA \,\%\, (an) + jan + \frac{ab \pm ab}{2} \tag{8.58}$$

Applying Lemma 5 yields

$$\frac{1}{aN'} \sum_{ij} c_{ij} = \frac{n(aN'-1) + ab \pm ab}{2} \tag{8.59}$$

Substituting this into (8.52) finalizes the derivation.

For (8.18), note that once $\hat{\mathbb{E}}[\alpha_k] = \frac{B+N-n}{2}$ is set, the largest relative error occurs if $\mathbb{E}[\alpha_k]$ is on the lower boundary of the possible set, i.e. if $\mathbb{E}[\alpha_k] = \frac{B+N-n-ab}{2}$. The distance between $\hat{\mathbb{E}}[\alpha_k]$ and $\mathbb{E}[\alpha_k]$ is then $\frac{ab}{2}$. Division yields the proposition.

The largest value of (α_k) depends on the largest value of c_{ij} which again can be bounded via the bounds of the ceiling operator:

$$\begin{aligned} \max_k \ \alpha_k &= \max_{ij} \ c_{ij} + (B'-1)ab \\ &= \max_i \ \{iA \,\%\, an\} + \max_j \ \{jan\} + B \\ &= n(a-1) + (N'-1)an + B = B + N - n \end{aligned} \tag{8.60}$$

which proves the corollary. □

Proof of Corollary 2

We separate the values generated by (ε_k) in $aB'N'$-long groups of elements. Then the expectation can be expressed via

$$\begin{aligned}\mathbb{E}[\varepsilon_k] = \lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K}\varepsilon_k &= \lim_{K\to\infty}\frac{1}{K}\sum_{k'=0}^{K}\frac{1}{aB'N'}\sum_{k=0}^{aB'N'-1}\varepsilon_{k'aB'N'+k}\\ &= \frac{1}{aB'N'}\sum_{k=0}^{aB'N'-1}\lim_{K\to\infty}\frac{1}{K}\sum_{k'=0}^{K}\varepsilon_{k'aB'N'+k} \\ &= \frac{1}{aB'N'}\sum_{k=0}^{aB'N'-1} p\varepsilon_k^{[0]} + p\bar{p}^1\varepsilon_k^{[1]} + p\bar{p}^2\varepsilon_k^{[2]} + \dots\end{aligned} \tag{8.61}$$

Notice that this way, we circumvented the "impossibility" of the assumption for $(\varepsilon_k^{[l]})$, since with probability $p\bar{p}^l$ it is indeed the case that, relative to time-step $t = kA$, the last l transmissions did fail and the $(l+1)$-th did succeed, for each of the $aB'N'$ elements. According to Theorem 7 and Lemma 7 we get

$$\begin{aligned}\frac{1}{aN'}\sum_{\substack{i=0,\dots a-1\\ j=0,\dots N'-1}} c_{ij}^{[l]} &= \frac{1}{aN'}\sum_{\substack{i=0,\dots a-1\\ j=0,\dots N'-1}}\left(2 + lN + jan + \frac{ab \pm ab}{2} + (iA - \Delta_N - 1)\,\%\,an\right)\\ &= 2 + lN + \frac{(N-n)}{2} + \frac{ab \pm ab}{2} + (\Delta_N + 1)\,\overline{\%}\,n\end{aligned} \tag{8.62}$$

because $(iA - \Delta_N - 1)\,\%\,an = (iA\,\%\,an - \Delta_N - 1)\,\%\,an$ which, over $i = 0,\dots a-1$, generates the set $(\langle 0, n, a\rangle - -\Delta_N - 1)\,\%\,an$. Hence, with $K = 2 + (\Delta_N + 1)\,\overline{\%}\,n$,

$$\begin{aligned}\frac{1}{aB'N'}\sum_{k=0}^{aB'N'}\varepsilon_k^{[l]} &= \frac{1}{aB'N'}\sum_{\substack{i=0,\dots a-1\\ j=0,\dots N'-1}}\left\langle c_{ij}^{[l]}, ab, B'\right\rangle\\ &= ab\frac{(B'-1)}{2} + \frac{1}{aN'}\sum_{\substack{i=0,\dots a-1\\ j=0,\dots N'-1}} c_{ij}^{[l]}\\ &= \frac{B+N-n}{2} + K + lN \pm \frac{ab}{2}\end{aligned} \tag{8.63}$$

Substituting this back into (8.61) and using the well-known results on polylogarithms yields

$$\begin{aligned}\mathbb{E}[\varepsilon_k] &= \frac{B+N-n}{2} + K \pm \frac{ab}{2} + p\lim_{L\to\infty}\sum_{l=0}^{L}\bar{p}^l lN\\ &= \frac{B+N-n}{2} + K \pm \frac{ab}{2} + \frac{\bar{p}}{p}N\end{aligned} \tag{8.64}$$

This proves proposition (8.23).

As for proposition (8.26), notice that (polylogarithm)

$$p\left(1 + \bar{p} + \bar{p}^2 + \dots + \bar{p}^{\lambda-1}\right) = 1 - \bar{p}^\lambda \tag{8.65}$$

I.e. considering only the first λ summands in (8.61) means considering only $1 - \bar{p}^\lambda$ percent of all possibly occurring values. The largest value among these belongs to the $\bar{p}^{\lambda-1}$ part and is easily evaluated to be

$$\max_{ij} c_{ij}^{[\lambda-1]} + (B'-1)ab \leq B + N - n + K + (\lambda-1)N \tag{8.66}$$

Setting $\sigma = 1 - \bar{p}^{\lambda}$ yields proposition (8.26).

Finally, (8.25) follows for the same reasons as the corresponding proposition from Corollary 1. □

Definition 3. *Given a set of primes $\mathcal{X} = \{x_1, x_2, \dots\}$ we define the set of all multiples generated by this set as*

$$\mathcal{M}_{\mathcal{X}} = \left\{ a : \ a = k \cdot \prod_{x \in \mathcal{X}'} x, \quad \begin{matrix} k \in \mathbb{Z} \\ \mathcal{X}' \subset \mathcal{X} \\ \mathcal{X}' \neq \varnothing \end{matrix} \right\} = \bigcup_{x \in \mathcal{X}} x\mathbb{Z} \tag{8.67}$$

Lemma 8. *Let $\mathcal{X} = \{x_1, x_2, \dots\}$ be a set of prime numbers x_i and $\mathcal{Y} = \{y\}$ be a singleton set with another prime $y \notin \mathcal{X}$. Denote the sets of all multiples generated by them with $\mathcal{M}_{\mathcal{X}}$ and $\mathcal{M}_{\mathcal{Y}}$, respectively. Then*

$$\frac{|\mathcal{M}_{\mathcal{X}}|}{|\mathbb{Z}|} = \frac{|\mathcal{M}_{\mathcal{X}} \cap \mathcal{M}_{\mathcal{Y}}|}{|\mathcal{M}_{\mathcal{Y}}|} \tag{8.68}$$

I.e. proportionally, there are as many multiples generated by $\mathcal{X}$ in $\mathbb{Z}$ as there are in $\mathcal{M}_{\mathcal{Y}}$.

Proof. If $a \in \mathcal{M}_{\mathcal{X}} \cap \mathcal{M}_{\mathcal{Y}}$ then

$$a = k \cdot \prod_{x \in \mathcal{X}'} x = k' \cdot y \tag{8.69}$$

for some $k, k' \in \mathbb{Z}$ and $\varnothing \neq \mathcal{X}' \subset \mathcal{X}$. However, since $y \notin \mathcal{X}$ it must be hold that

$$a = k'' \cdot y \cdot \prod_{x \in \mathcal{X}'} x, \qquad k'' \in \mathbb{Z} \tag{8.70}$$

but then

$$\mathcal{M}_{\mathcal{X}} \cap \mathcal{M}_{\mathcal{Y}} = \left\{ ya : \quad a = k \cdot \prod_{x \in \mathcal{X}'} x, \quad \begin{matrix} k \in \mathbb{Z} \\ \mathcal{X}' \subset \mathcal{X} \\ \mathcal{X}' \neq \varnothing \end{matrix} \right\} \tag{8.71}$$

I.e. $\mathcal{M}_{\mathcal{X}} \cap \mathcal{M}_{\mathcal{Y}} = y\mathcal{M}_{\mathcal{X}}$. Therefore we have $\frac{|\mathcal{M}_{\mathcal{X}} \cap \mathcal{M}_{\mathcal{Y}}|}{|\mathcal{M}_{\mathcal{X}}|} = y = \frac{|y\mathbb{Z}|}{|\mathbb{Z}|} = \frac{|\mathcal{M}_{\mathcal{Y}}|}{|\mathbb{Z}|}$. □

Lemma 9. *Let R be a random integer uniformly drawn from $\mathbb{N}_+$. Further let $M = \prod_{x \in \mathcal{X}} x^{r(x)}$ be an integer together with its prime factorization. Then the probability that M and R are coprime is*

$$\mathbb{P}\left[\gcd\left(R, \prod_{x \in \mathcal{X}} x^{r(x)} \right) = 1 \right] = \sum_{\mathcal{X}' \subset \mathcal{X}} \frac{(-1)^{|\mathcal{X}'|}}{\prod\limits_{x \in \mathcal{X}'} x} \tag{8.72}$$

Proof. First suppose that $r(x) = 1$ for all $x \in \mathcal{X}$. We employ an inductive reasoning:

If $M = x$ then in agreement with (8.72) we have

$$\mathbb{P}[\gcd(R, x) = 1] = 1 - \frac{1}{x} = \frac{(-1)^0}{1} + \frac{(-1)^1}{x} \tag{8.73}$$

which obviously is correct.

Next suppose that the claim holds for some $M = \prod_{x \in \mathcal{X}} x$ and note that the following relation holds

$$\mathbb{P}\left[\gcd\left(R, \prod_{x \in \mathcal{X}} x\right) = 1\right] = 1 - \mathbb{P}[R \in \mathcal{M}_{\mathcal{X}}] = \theta \tag{8.74}$$

where θ is a placeholder for (8.72). When we expand M by an additional prime $y \notin \mathcal{X}$ (and for convenience define the singleton set $\mathcal{Y} = \{y\}$), we get

$$\begin{aligned}
&\mathbb{P}\left[\gcd\left(R, y \cdot \prod_{x \in \mathcal{X}} x\right) = 1\right] = 1 - \mathbb{P}[R \in \mathcal{M}_{\mathcal{X} \cup \mathcal{Y}}] \\
&= 1 - \mathbb{P}[R \in \mathcal{M}_{\mathcal{X}} \cup \mathcal{M}_{\mathcal{Y}}] \\
&= 1 - \mathbb{P}[R \in \mathcal{M}_{\mathcal{X}}] - \mathbb{P}[R \in \mathcal{M}_{\mathcal{Y}}] + \mathbb{P}[R \in \mathcal{M}_{\mathcal{X}} \cap \mathcal{M}_{\mathcal{Y}}] \\
&= \theta - \frac{1}{y} + \frac{|\mathcal{M}_{\mathcal{X}} \cap \mathcal{M}_{\mathcal{Y}}|}{|\mathbb{Z}|} \\
&\overset{\text{Lemma 8}}{=} \theta - \frac{1}{y} + \frac{|\mathcal{M}_{\mathcal{X}}|\,|\mathcal{M}_{\mathcal{Y}}|}{|\mathbb{Z}|\,|\mathbb{Z}|} = \theta - \frac{1}{y} + (1-\theta)\frac{1}{y} = \theta\left(1 - \frac{1}{y}\right)
\end{aligned} \tag{8.75}$$

Substituting for θ and defining $\mathcal{Z} = \mathcal{X} \cup \{y\}$ we obtain

$$\begin{aligned}
\mathbb{P}\left[\gcd\left(R, y \cdot \prod_{x \in \mathcal{X}} x\right) = 1\right] &= \sum_{\mathcal{X}' \subset \mathcal{X}} \frac{(-1)^{|\mathcal{X}'|}}{\prod_{x \in \mathcal{X}'} x} - \frac{1}{y} \sum_{\mathcal{X}' \subset \mathcal{X}} \frac{(-1)^{|\mathcal{X}'|}}{\prod_{x \in \mathcal{X}'} x} \\
= \sum_{\substack{\mathcal{Z}' \subset \mathcal{Z} \\ y \notin \mathcal{Z}'}} \frac{(-1)^{|\mathcal{Z}'|}}{\prod_{x \in \mathcal{Z}'} x} &+ \sum_{\substack{\mathcal{Z}' \subset \mathcal{Z} \\ y \notin \mathcal{Z}'}} \frac{(-1)^{|\mathcal{Z}'|+1}}{y \prod_{x \in \mathcal{Z}'} x} = \sum_{\mathcal{Z}' \subset \mathcal{Z}} \frac{(-1)^{|\mathcal{Z}'|}}{\prod_{z \in \mathcal{Z}'} z}
\end{aligned} \tag{8.76}$$

Per induction, this proves the claim if $r(x) = 1$ for all $x \in \mathcal{X}$.

This leaves us to show that claim (8.72) is indeed independent of the multiplicity of the prime factors. This can be done by construction. We have

$$\begin{aligned}
\mathbb{P}\left[\gcd\left(R, \prod_{x \in \mathcal{X}} x^{r(x)}\right) = 1\right] &= \mathbb{P}\left[\gcd\left(R, \prod_{x \in \mathcal{X}} x^{r(x)-1} \cdot \prod_{x \in \mathcal{X}} x\right) = 1\right] \\
&= \mathbb{P}\left[\gcd\left(R, y \cdot \prod_{x \in \mathcal{X}} x\right) = 1\right] = 1 - \mathbb{P}[R \in \mathcal{M}_{\mathcal{X}} \cup \mathcal{M}_{\mathcal{Y}}] \\
&= 1 - \mathbb{P}[R \in \mathcal{M}_{\mathcal{X}}] = \mathbb{P}\left[\gcd\left(R, \prod_{x \in \mathcal{X}} x\right) = 1\right]
\end{aligned} \tag{8.77}$$

where this time $y = \prod_{x \in \mathcal{X}} x^{r(x)-1} \in \mathcal{M}_{\mathcal{X}}$ such that $\mathcal{M}_{\mathcal{Y}} \subset \mathcal{M}_{\mathcal{X}}$. This completes the proof. □

Lemma 10. *Let R be a random integer uniformly drawn from $\mathbb{N}_+$. Further let $M = \prod_{x\in\mathcal{X}} x$ be an integer together with its prime factorization, where $\mathcal{X} = \{x_1, x_2, \dots\}$ is a multiset of primes (i.e. a single prime can be contained more than once, corresponding to its multiplicity in M). Let further $\mathcal{X}' \subset \mathcal{X}$ such that $D = \prod_{x\in\mathcal{X}'} x$ defines a divisor of M. Then the probability that D is the greatest common divisor between R and M is*

$$\mathbb{P}[\gcd(R,M) = D] = \frac{1}{D}\mathbb{P}\left[\gcd\left(R, \frac{M}{D}\right) = 1\right] \tag{8.78}$$

Proof. We have the straightforward construction

$$\begin{aligned}\mathbb{P}[\gcd(R,M) = D] &= \mathbb{P}\left[R \in \mathcal{M}_D \cap \left(\bigcap_{x\in\mathcal{X}\setminus\mathcal{X}'} R \notin \mathcal{M}_{x\cdot D}\right)\right] \\ &= \mathbb{P}\left[\bigcap_{x\in\mathcal{X}\setminus\mathcal{X}'} R \notin \mathcal{M}_{x\cdot D} \,\middle|\, R \in \mathcal{M}_D\right]\mathbb{P}[R \in \mathcal{M}_D] \\ &= \mathbb{P}\left[\bigcap_{x\in\mathcal{X}\setminus\mathcal{X}'} R \notin \mathcal{M}_x\right]\mathbb{P}[R \in \mathcal{M}_D] \\ &= \left(1 - \mathbb{P}\left[\bigcup_{x\in\mathcal{X}\setminus\mathcal{X}'} R \in \mathcal{M}_x\right]\right)\mathbb{P}[R \in \mathcal{M}_D] \\ &= \left(1 - \mathbb{P}\left[R \in \mathcal{M}_{\mathcal{X}\setminus\mathcal{X}'}\right]\right)\mathbb{P}[R \in \mathcal{M}_D] \\ &= \frac{1}{D}\mathbb{P}\left[\gcd\left(R, \frac{\prod\limits_{x\in\mathcal{X}} x}{\prod\limits_{x\in\mathcal{X}'} x}\right) = 1\right]\end{aligned} \tag{8.79}$$

Substituting M and D yields the proposition. □

Proof of Theorem 8
Define $\pi(\mathcal{X}, r(\mathcal{X})) := \prod_{x\in\mathcal{X}} x^{r(x)}$ as an efficient way to denote a prime factorization where $\mathcal{X}$ is the set of primes and $r(\mathcal{X})$ is the vector of corresponding multiplicities. Then we have

$$\mathbb{E}[\gcd(R,M)] = \mathbb{E}[\gcd(R, \pi(\mathcal{X}, r(\mathcal{X})))]$$

$$= \sum_{\mathcal{X}'\subset\mathcal{X}} \sum_{r'\le r(\mathcal{X}')} \pi(\mathcal{X}', r') \cdot \mathbb{P}[\gcd(R, \pi(\mathcal{X}, r(\mathcal{X}))) = \pi(\mathcal{X}', r')] \tag{8.80}$$

$$\overset{\text{Lemma 10}}{=} \sum_{\mathcal{X}'\subset\mathcal{X}} \sum_{r'\le r(\mathcal{X}')} \mathbb{P}\left[\gcd\left(R, \frac{\pi(\mathcal{X}, r(\mathcal{X}))}{\pi(\mathcal{X}', r')}\right) = 1\right] \tag{8.81}$$

$$= \sum_{\mathcal{X}'\subset\mathcal{X}} \sum_{r'\le r(\mathcal{X}')} \mathbb{P}[\gcd(R, \pi(\mathcal{X}', r')) = 1] \tag{8.82}$$

$$= \sum_{\mathcal{X}'\subset\mathcal{X}} \prod_{x\in\mathcal{X}'} r(x) \cdot \mathbb{P}\left[\gcd\left(R, \prod_{x\in\mathcal{X}'} x\right) = 1\right] \tag{8.83}$$

where the vector r' must be greater or equal to 1 in every component. Equation (8.82) follows by symmetry: the set of all divisors of a given number M is the same as the set of quotients obtained by dividing M by each of its divisors separately. Equation (8.83) follows by the independence of multiplicity (established in Lemma 9) and the fact that $\prod_{x\in\mathcal{X}'} r(x) = \sum_{r'\leq r(\mathcal{X}')} 1$. Applying Lemma 9 then yields the proposition. $\square$

8.3 Concluding Remarks

Obviously, the proof in the appendix is the main contribution of the paper. While it is only presented through rigorous formulas, there exists quite a rich geometric interpretation for the main ideas behind it which we do not want to withhold.

We start with the term (8.41), the evolution of the AoI:

$$\eta_t + (\beta_t - \eta_t)\,\%\,B \tag{8.84}$$

In order to obtain the distribution of the AoI sequence, we have Lemma 5 stating that both η_t and β_t generate sets of equally distanced values. However, the key lies in not treating both sequences equally but rather imagining η_t as some fixed disturbance that acts on the set generated by β_t. Figure 8.7 illustrates this case. Starting from the set generated by β_t on the top, in a first step, η_t shifts those values towards the negative part of the number line. After that, by virtue of the modulo-operator, all negative values are relocated back into the original range (which was $[0, B]$) while leaving the separation property of the values intact. Finally, η_t again shifts all values, but this time in the opposite direction. The details in the proof deal with figuring out which values from β_t encounter which values from η_t and where exactly the sets are shifted to.

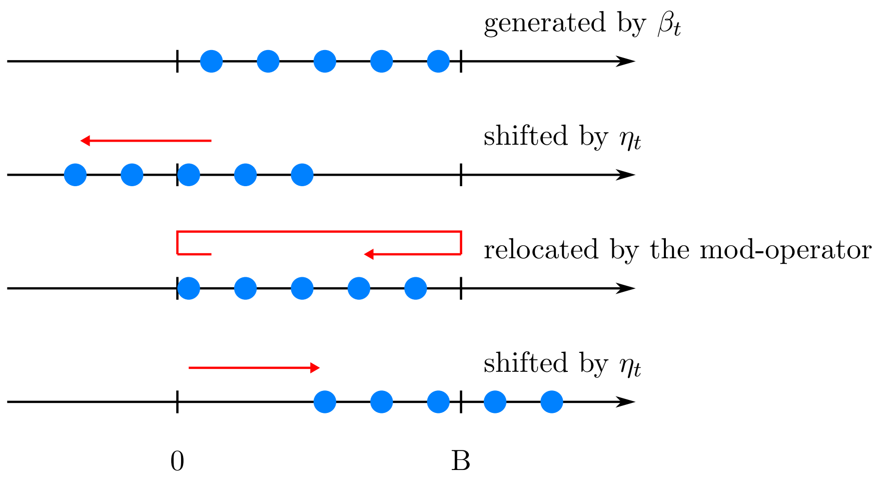


Fig. 8.7: Visualization of the proof for Theorem 6. A single disturbance η_t acts on a set of values, generated by β_t via equation (8.84).

For the second major proof, which deals with the expectation of the gcd, there exist visual representations of Lemma 8 and Lemma 9. Lemma 8 is concerned with the relative occurrence of multiples of a prime on the number line, and states that it stays the same when observing the number line through a grid which is also generated by the multiples of a prime. This finds a neat illustration in Figure 8.8. Here, the horizontal grids represent the number lines. In the first grid, all multiples of 2 are marked in blue, as are multiples of 3 in the second grid. (Of course, those grids each only present an exemplary finite part of the infinite grid.) The third line collects all numbers from the second grid that coincide with a multiple of 2 (i.e. with a blue entry in the first grid). If all these numbers are rearranged to form a new grid (the fourth grid), it can be observed that the amount of blue entries, relative to the number of white entries, is again $\frac{1}{3}$, i.e. the same as it was before in the second grid. In other words, even if we observe only every second number on the number line, it is still true that every third of these numbers is a multiple of 3.

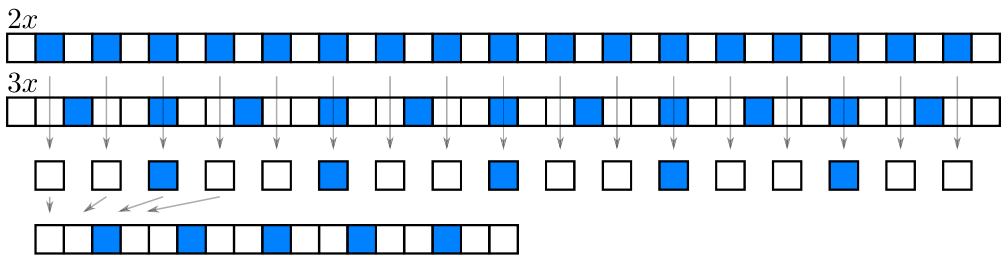


Fig. 8.8: Visualization of Lemma 8. Observing the number line through a mask, generated by all multiples of a prime, does not change the relative occurrence of multiples of other primes.

Through similar means, Figure 8.9 illustrates Lemma 9 that gives a formula for the probability of the gcd being 1. In this example, we ask for the probability of the gcd between a random number and $30 = (2 \cdot 3 \cdot 5)$ to be 1. This is the same as the probability of these two numbers being coprime. To find the answer, we mark all numbers that are not coprime on the number line/grid and simply count them. Obviously, all numbers that are multiples of 2 or 3 or 5 cannot be coprime to 30. These numbers are marked with blue in the first three grids. We find that $\frac{1}{2}$ of all numbers are multiples of 2, and that $\frac{1}{3}$ of all numbers are multiples of 3, and that $\frac{1}{5}$ of all numbers are multiples of 5. However adding all these fractions overestimates the amount of numbers that are multiples of 2,3 or 5, since some numbers are accounted for more than once. This is illustrated in the fourth grid where we marked all previously marked numbers. However, instead of using the color blue, we gave each number a counter that counts in how many grids this number was marked. E.g. 6 was marked both in the grid with multiples of 2 as well as in the grid with multiples of 3.

To compensate for the increased occurrence of such values, we have to deduct $\frac{1}{2\cdot 3}$ of all numbers, and $\frac{1}{2\cdot 5}$ of all numbers and $\frac{1}{3\cdot 5}$ of all numbers from the overestimation $\frac{1}{2}+\frac{1}{3}+\frac{1}{5}$. This is illustrated by the next 4 grids, who mark corresponding values and deduct them from the previous "sum". Evidently, now almost every number that is a multiple of 2,3 or 5 has been considered exactly once, except 30 and multiples of it, which are not considered at all anymore. Again, to compensate we have to add the probability of all numbers, that are a multiple of 30, i.e. $\frac{1}{2\cdot 35}$, illustrated in the last two grids. In the end, we end up considering every multiple of 2,3 and 5 exactly once, if we execute all these compensations to the overestimate that we started with which gives us

$$\mathbb{P}[\gcd(R, 2 \cdot 3 \cdot 5) = 1] = \frac{1}{2} + \frac{1}{3} + \frac{1}{5} - \frac{1}{2 \cdot 3} - \frac{1}{2 \cdot 5} - \frac{1}{3 \cdot 5} + \frac{1}{2 \cdot 3 \cdot 5} \tag{8.85}$$

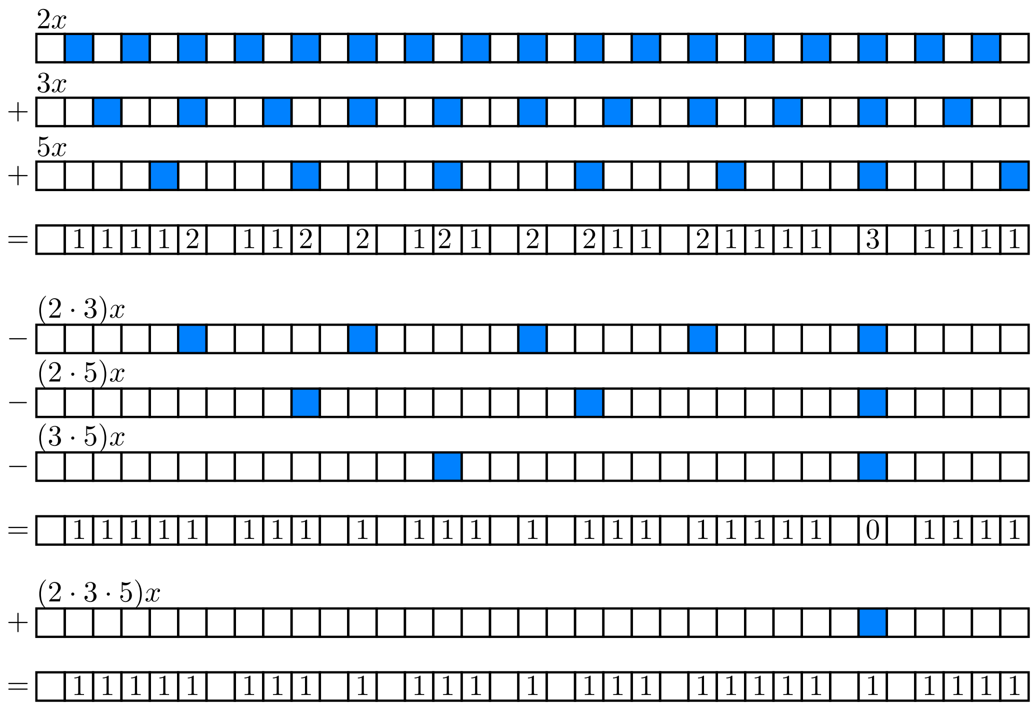


Fig. 8.9: Visualization of Lemma 9. The observed structure in (8.85) stems from an iterative scheme to compensate for multiplicities.

Abstract in German Language

Im Rahmen dieser Dissertation wurden die folgenden 3 Schwerpunkte bearbeitet:

- Zum einen wurde eine prädiktive Methode zur Kontrolle von Warteschlangennetzwerken entwickelt. Unter Annahme eines beschreibbaren stochastischen Prozesses, der die Übertragungswahrscheinlichkeit von Warte- zu Warteschlange beschreibt, ist diese Methode eine direkte Erweiterung der allseits bekannten "Back-Pressure" Methoden auf Prädiktionshorizonte variabler Länge. Durch Verbindung von Beweismethoden aus den Gebieten der modellprädiktiven Regelungstheorie und der Warteschlangentheorie konnte bewiesen werden, dass sich diese Methode optimal bezüglich des Durchflusses verhält. Außerdem konnte herausgearbeitet werden, warum diese Methode weiterführende Netzwerkstrukturen stabilisiert, die im Gegensatz zu herkömmlichen Strukturen auch synchronisierte Warteschlangen enthält. In diesem Zusammenhang konnte auch eine Verbindung der Stabilitätseigenschaften solcher synchronisierter Warteschlangen mit Zufallsbewegungen auf Faktorräumen hergestellt werden.

- Des Weiteren wurden 2 Algorithmen entwickelt, die es erlauben, Kommunikationsverzögerungen durch geschickte Regelung des Netzwerkes verlässlich vorherzusagen. Beide Algorithmen arbeiten nach den Prinzipien der modellprädiktiven Regelung, unterscheiden sich jedoch insbesondere hinsichtlich ihrer Zielfunktion. Während die Optimierung bei beiden auf ein kombinatorisches Problem hinausläuft (was durch die diskrete Funktionsweise des Netzwerkes nicht verhindert werden kann), konnten wir die Komplexität einer der Algorithmen auf eine lineare Zielfunktion beschränken. Dies ermöglicht eine bessere Skalierung mit den Systemparametern im Gegensatz zu den üblichen quadratischen Zielfunktionen.

- Als Letztes wurde die Prädiktion des Age-of-Information untersucht. Resultate waren hier ein Algorithmus zur verlässlichen Vorhersage von AoI-Werten in Netzwerken mit flacher Topologie und eine Methode zur Berechnung der exakten Wahrscheinlichkeitsverteilung über dem AoI-Zustandsraum. Außerdem wurde die Verteilung der AoI-Werte in Agenten eines "Wireless Token Ring Protokolls" hergeleitet. Dies ermöglicht die Ermittlung aller relevanten Momente des stochastischen Prozesses, der das AoI beschreibt und somit eine Vorhersage des AoI im stochastischen Sinne.

Bibliography

[1] Guillermo Pocovi, Beatriz Soret, Klaus I. Pedersen, and Preben Mogensen. Mac layer enhancements for ultra-reliable low-latency communications in cellular networks. *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017.

[2] Anthony Ephremides. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Transactions on Automatic Control*, 1992.

[3] Sean Meyn. Stability and Asymptotic Optimality of Generalized MaxWeight Policies. *SIAM Journal on Control and Optimization*, 2009.

[4] Martin Kasparick and Gerhard Wunder. Stable wireless network control under service constraints. *IEEE Transactions on Control of Network Systems*, 2018.

[5] Sean Meyn. *Control Techniques for Complex Networks.* Springer, 2011.

[6] Vijay G. Subramanian and Douglas J. Leith. Draining time based scheduling algorithm. In *46th IEEE Conference on Decision and Control*, 2007.

[7] Gerhard Wunder and Chan Zhou. Queueing analysis for the OFDMA downlink: Throughput regions, delay and exponential backlog bounds. *IEEE Transactions on Wireless Communications*, 2009.

[8] Chan Zhou and Gerhard Wunder. Throughput-optimal scheduling with low average delay for cellular broadcast systems. *Eurasip Journal on Advances in Signal Processing*, 2009.

[9] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 2000.

[10] M.D. Fragoso O.L.V. Costa and R.P. Marques. *Discrete-Time Markov Jump Linear Systems.* Springer, 2005.

[11] Nick McKeown, Adisak Mekkittikul, Venkat Anantharam, and Jean Walrand. Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 1999.

[12] Michael J. Neely, Eytan Modiano, and Charles E. Rohrs. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Communications*, 2005.

[13] Wajahat Khan, Long Bao Le, and Eytan Modiano. Autonomous routing algorithms for networks with wide-spread failures. In *IEEE Military Communications Conference MILCOM*, 2009.

[14] Lei Ying, Sanjay Shakkottai, Aneesh Reddy, and Shihuan Liu. On combining shortest-path and back-pressure routing over multihop wireless networks. *IEEE/ACM Transactions on Networking*, 2011.

[15] Po Kai Huang, Xiaojun Lin, and Chih Chun Wang. A low-complexity congestion control and scheduling algorithm for multihop wireless networks with order-optimal per-flow delay. *IEEE/ACM Transactions on Networking*, 2013.

[16] Haozhi Xiong, Ruogu Li, Atilla Eryilmaz, and Eylem Ekici. Delay-aware cross-layer design for network utility maximization in multi-hop networks. *IEEE Journal on Selected Areas in Communications*, 2011.

[17] Johan S.H. Van Leeuwaarden, Erjen Lefeber, Yoni Nazarathy, and Jacobus E. Rooda. Model Predictive Control for the acquisition queue and related queueing networks. *5th International Conference on Queueing Theory and Network Applications, QTNA*, 2010.

[18] David Guzman, Richard Schoeffauer, and Gerhard Wunder. Predictive network control in multi-connectivity mobility for URLLC services. *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, 2019.

[19] Byung Gun Park and Wook Hyun Kwon. Robust one-step receding horizon control of discrete-time Markovian jump uncertain systems. *Automatica*, 2002.

[20] Shaikshavali Chitraganti, Samir Aberkane, Christophe Aubrun, Guillermo Valencia-Palomo, and Vasile Dragan. On control of discrete-time state-dependent jump linear systems with probabilistic constraints: A receding horizon approach. *Systems and Control Letters*, 2014.

[21] Jens Tonne and Olaf Stursberg. Constraint robust model predictive control for jump Markov linear systems with additive disturbances. *European Control Conference, ECC*, 2017.

[22] Yanqing Liu, Yanyan Yin, Fei Liu, and Kok Lay Teo. Constrained MPC design of nonlinear Markov jump system with nonhomogeneous process. *Nonlinear Analysis: Hybrid Systems*, 2015.

[23] Jens Tonne and Olaf Stursberg. Fast Robust Model Predictive Control for Nonlinear Jump Markov Systems. *IFAC-PapersOnLine*, 2017.

[24] F. G. Foster. Markoff chains with an enumerable number of states and a class of cascade processes. *Mathematical Proceedings of the Cambridge Philosophical Society*, 1951.

[25] Richard Schoeffauer and Gerhard Wunder. Predictive network control and throughput sub-optimality of max weight. *European Conference on Networks and Communications (EuCNC)*, 2018.

[26] Richard Schoeffauer and Gerhard Wunder. A Linear Algorithm for Reliable Predictive Network Control. *IEEE Globecom Workshops, GC Wkshps*, 2019.

[27] Pierre Bremaud. Markov chains. *Springer*, 1975.

[28] U. Narayan Bhat. Finite capacity assembly-like queues. *Queueing Systems*, 1986.

[29] E. H. Lipper and B. Sengupta. Assembly-like queues with finite capacity: Bounds, asymptotics and approximations. *Queueing Systems*, 1986.

[30] Wallace J. Hopp and John T. Simon. Bounds and heuristics for assembly-like queues. *Queueing Systems*, 1989.

[31] Stanley B. Gershwin. Assembly/Disassembly Systems: An Efficient Decomposition Algorithm For Tree-Structured Networks. *IIE Transactions (Institute of Industrial Engineers)*, 1991.

[32] Keun Chae Jeong and Yeong Dae Kim. Performance analysis of assembly/disassembly systems with unreliable machines and random processing times. *IIE Transactions (Institute of Industrial Engineers)*, 1998.

[33] Michael Manitz. Queueing-model based analysis of assembly lines with finite buffers and general service times. *Computers and Operations Research*, 2008.

[34] Erica L. Plambeck and Amy R. Ward. Optimal control of a high-volume assemble-to-order system with maximum leadtime quotation and expediting. *Queueing Systems*, 2008.

[35] Itai Gurvich and Amy Ward. On the dynamic control of matching queues. *Stochastic Systems*, 2014.

[36] Eline De Cuypere, Koen De Turck, and Dieter Fiems. A Maclaurin-series expansion approach to multiple paired queues. *Operations Research Letters*, 2014.

[37] Mariana Olvera-Cravioto and Octavio Ruiz-Lacedelli. Parallel queues with synchronization. *arXiv*, 2014.

[38] Burak Büke and Hanyi Chen. Stabilizing policies for probabilistic matching systems. *Queueing Systems*, 2015.

[39] Mehmet Fatih Aktas and Emina Soljanin. Anonymity Mixes as (Partial) Assembly Queues: Modeling and Analysis. *IEEE Information Theory Workshop, ITW*, 2019.

[40] J . Michael Harrison. Assembly-like Queues. *Journal of Applied Probability*, 1973.

[41] Guy Latouche. Queues With Paired Customers. *Journal of Applied Probability*, 1981.

[42] David G. Kendall. On non-dissipative Markoff chains with an enumerable infinity of states. *Mathematical Proceedings of the Cambridge Philosophical Society*, 1951.

[43] F. G. Foster. On the Stochastic Matrices Associated with Certain Queuing Processes. *The Annals of Mathematical Statistics*, 1953.

[44] Georg Pólya. Über eine Aufgabe der Wahrscheinlichkeitsrechnung betreffend die Irrfahrt im Straßennetz. *Mathematische Annalen*, 1921.

[45] Edward A. Lee. The past, present and future of cyber-physical systems: A focus on models. *Sensors (Switzerland)*, 2015.

[46] Mohamed El Mongi Ben Gaid, Arben Çela, and Yskandar Hamam. Optimal integrated control and scheduling of networked control systems with communication constraints: Application to a car suspension system. *IEEE Transactions on Control Systems Technology*, 2006.

[47] Gregory C. Walsh, Hong Ye, and Linda G. Bushnell. Stability analysis of networked control systems. *IEEE Transactions on Control Systems Technology*, 2002.

[48] Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. Co-design of cyber-physical systems via controllers with flexible delay constraints. *16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2011.

[49] Steffen Linsenmayer, Dimos V. Dimarogonas, and Frank Allgöwer. Event-Based Vehicle Coordination Using Nonlinear Unidirectional Controllers. *IEEE Transactions on Control of Network Systems*, 2018.

[50] Dominic Gross and Olaf Stursberg. A cooperative distributed MPC algorithm with event-based communication and parallel optimization. *IEEE Transactions on Control of Network Systems*, 2016.

[51] Fabian Mager, Dominik Baumann, Romain Jacob, Lothar Thiele, Sebastian Trimpe, and Marco Zimmerling. Feedback control goes wireless. *ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS*, 2019.

[52] William B. Dunbar and Richard M. Murray. Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 2006.

[53] Elisa Franco, Lalo Magni, Thomas Parisini, Marios M. Polycarpou, and Davide M. Raimondo. Cooperative constrained control of distributed agents with nonlinear dynamics and delayed information exchange: A stabilizing receding-horizon approach. *IEEE Transactions on Automatic Control*, 2008.

[54] D. Groß and O. Stursberg. Distributed predictive control of communicating and constrained systems. *ZAMM Zeitschrift fur Angewandte Mathematik und Mechanik*, 2014.

[55] Zhuo Zhang, Yang Shi, Zexu Zhang, Hui Zhang, and Sheng Bi. Modified Order-Reduction Method for Distributed Control of Multi-Spacecraft Networks with Time-Varying Delays. *IEEE Transactions on Control of Network Systems*, 2018.

[56] Paul J. Goulart, Eric C. Kerrigan, and Jan M. MacIejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 2006.

[57] J. Hahn, Richard Schoeffauer, G. Wunder, and O. Stursberg. Distributed MPC with Prediction of Time-Varying Communication Delay. *IFAC-PapersOnLine*, 2018.

[58] Gerhard Wunder and Thomas Michel. Optimal resource allocation for parallel Gaussian broadcast channels: Minimum rate constraints and sum power minimization. *IEEE Transactions on Information Theory*, 2007.

[59] Antonio Frangioni, Laura Galli, and Giovanni Stea. Delay-constrained routing problems: Accurate scheduling models and admission control. *Computers and Operations Research*, 2017.

[60] Juyul Lee and Nihar Jindal. Delay constrained scheduling over fading channels: Optimal policies for monomial energy-cost functions. *IEEE International Conference on Communications*, 2009.

[61] Elias Munapo. Solving the Binary Linear Programming Model in Polynomial Time. *American Journal of Operations Research*, 2016.

[62] Edward A. Lee. Cyber physical systems: Design challenges. *11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC*, 2008.

[63] Patricia Derler, Edward A. Lee, and Alberto Sangiovanni Vincentelli. Modeling cyber–physical systems. *Proceedings of the IEEE*, 2012.

[64] Lars Grüne and Jürgen Pannek. Nonlinear model predictive control. In *Nonlinear model predictive control*. Springer, 2017.

[65] J.B. Rawlings, E.S. Meadows, and K.R. Muske. Nonlinear Model Predictive Control: A Tutorial and Survey. *IFAC Proceedings Volumes*, 1994.

[66] Y. I. Lee and B. Kouvaritakis. A linear programming approach to constrained robust predictive control. *IEEE Transactions on Automatic Control*, 2000.

[67] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 2005.

[68] Daniele Bernardini and Alberto Bemporad. Stabilizing model predictive control of stochastic constrained linear systems. *IEEE Transactions on Automatic Control*, 2012.

[69] Giuseppe C. Calafiore and Lorenzo Fagiano. Robust model predictive control via scenario optimization. *IEEE Transactions on Automatic Control*, 2013.

[70] Yang Wang and Stephen Boys. Fast Model Predictive Control Using Online Optimization. *IEEE Transactions on Control Systems Technology*, 2010.

[71] Yu Gao and Kil To Chong. The explicit constrained min-max model predictive control of a discrete-time linear system with uncertain disturbances. *IEEE Transactions on Automatic Control*, 2012.

[72] Melanie Nicole Zeilinger, Colin Neil Jones, and Manfred Morari. Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization. *IEEE Transactions on Automatic Control*, 2011.

[73] Brett T. Stewart, Aswin N. Venkat, James B. Rawlings, Stephen J. Wright, and Gabriele Pannocchia. Cooperative distributed model predictive control. *Systems and Control Letters*, 2010.

[74] Panagiotis D. Christofides, Riccardo Scattolini, David Muñoz de la Peña, and Jinfeng Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers and Chemical Engineering*, 2013.

[75] Eduardo Camponogara, Dong Jia, Bruce H. Krogh, and Sarosh Talukdar. Distributed Model Predictive Control. *IEEE Control Systems Magazine*, 2002.

[76] Guo Ping Liu, Yuanqing Xia, Jie Chen, David Rees, and Wenshan Hu. Networked predictive control of systems with random network delays in both forward and feedback channels. *IEEE Transactions on Industrial Electronics*, 2007.

[77] Lars Grüne, Frank Allgöwer, Rolf Findeisen, Jörg Fischer, Dominic Groß, Uwe D Hanebeck, Benjamin Kern, Matthias A Müller, Jürgen Pannek, Marcus Reble, et al. Distributed and networked model predictive control. In *Control Theory of Digitally Networked Dynamic Systems*. Springer, 2014.

[78] Shengling Shi and Mircea Lazar. On distributed model predictive control for vehicle platooning with a recursive feasibility guarantee. *IFAC-PapersOnLine*, 2017.

[79] Svenja Blasi, Markus Kögel, and Rolf Findeisen. Distributed Model Predictive Control Using Cooperative Contract Options. *IFAC-PapersOnLine*, 2018.

[80] Nagacharan Teja Tangirala, Anuj Abraham, Apratim Choudhury, Pranjal Vyas, Rongkai Zhang, and Justin Dauwels. Analysis of Packet drops and Channel Crowding in Vehicle Platooning using V2X communication. *IEEE Symposium Series on Computational Intelligence, SSCI*, 2018.

[81] Dongyao Jia, Kejie Lu, and Jianping Wang. On the network connectivity of platoon-based vehicular cyber-physical systems. *Transportation Research Part C: Emerging Technologies*, 2014.

[82] Mohammad Hosseinzadeh Yamchi and Reza Mahboobi Esfanjani. Distributed predictive formation control of networked mobile robots subject to communication delay. *Robotics and Autonomous Systems*, 2017.

[83] Qi Wang, Katia Jaffrès-Runser, Jean Luc Scharbarg, Christian Fraboul, Yi Sun, Jun Li, and Zhongcheng Li. A thorough analysis of the performance of delay distribution models for IEEE 802.11 DCF. *Ad Hoc Networks*, 2015.

[84] Jing Wang, Jian Tang, Zhiyuan Xu, Yanzhi Wang, Guoliang Xue, Xing Zhang, and Dejun Yang. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. *IEEE INFOCOM*, 2017.

[85] Chen Qiu, Yanyan Zhang, Zhiyong Feng, Ping Zhang, and Shuguang Cui. Spatio-Temporal Wireless Traffic Prediction with Recurrent Neural Network. *IEEE Wireless Communications Letters*, 2018.

[86] Devarpita Sinha and Rajarshi Roy. Scheduling Status Update for Optimizing Age of Information in the Context of Industrial Cyber-Physical System. *IEEE Access*, 2019.

[87] Qing He, Di Yuan, and Anthony Ephremides. Optimal Link Scheduling for Age Minimization in Wireless Systems. *IEEE Transactions on Information Theory*, 2018.

[88] Yu Pin Hsu, Eytan Modiano, and Lingjie Duan. Scheduling Algorithms for Minimizing Age of Information in Wireless Broadcast Networks with Random Arrivals. *IEEE Transactions on Mobile Computing*, 2020.

[89] Igor Kadota, Abhishek Sinha, and Eytan Modiano. Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints. *IEEE/ACM Transactions on Networking*, 2019.

[90] M. Lazar, D. Muñoz de la Peña, W. P.M.H. Heemels, and T. Alamo. On input-to-state stability of min-max nonlinear model predictive control. *Systems and Control Letters*, 2008.

[91] Dominic Groß and Olaf Stursberg. A relaxed lyapunov condition for input-to-state stability of discrete-time nonlinear systems. *55th IEEE Conference on Decision and Control, CDC*, 2016.

[92] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems.* Cambridge University Press, 2017.

[93] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, 2018.

[94] Igor Kadota, Elif Uysal-Biyikoglu, Rahul Singh, and Eytan Modiano. Minimizing the Age of Information in Broadcast Wireless Networks. *54th Annual Allerton Conference on Communication, Control, and Computing*, 2016.

[95] Yu Pin Hsu, Eytan Modiano, and Lingjie Duan. Age of information: Design and analysis of optimal scheduling algorithms. *IEEE International Symposium on Information Theory*, 2017.

[96] Igor Kadota, Abhishek Sinha, Elif Uysal-Biyikoglu, Rahul Singh, and Eytan Modiano. Scheduling policies for minimizing age of information in broadcast wireless networks. *IEEE/ACM Transactions on Networking*, 2018.

[97] Igor Kadota, Abhishek Sinha, and Eytan Modiano. Optimizing Age of Information in Wireless Networks with Throughput Constraints. *IEEE INFOCOM*, 2018.

[98] Ahmed Arafa and Sennur Ulukus. Age-minimal transmission in energy harvesting two-hop networks. *IEEE Global Communications Conference, GLOBECOM*, 2017.

[99] Roy D. Yates. Age of information in a network of preemptive servers. *IEEE Conference on Computer Communications Workshops, INFOCOM*, 2018.

[100] Shahab Farazi, Andrew G. Klein, and D. Richard Brown. Fundamental bounds on the age of information in multi-hop global status update networks. *Journal of Communications and Networks*, 2019.

[101] Jannik Hahn, Richard Schoeffauer, Gerhard Wunder, and Olaf Stursberg. Using AoI Forecasts in Communicating and Robust Distributed Model-Predictive Control. *IEEE Transactions on Control of Network Systems*, 2021.

[102] Onur Ayan, Mikhail Vilgelm, Markus Klügel, Sandra Hirche, and Wolfgang Kellerer. Age-of-information vs. Value-of-information scheduling for cellular networked control systems. *arXiv*, 2019.

[103] Petr Kundrát. On the Asymptotics of the Difference Equation with a Proportional Delay. *Opuscula Mathematica*, 2006.

[104] J.R. Ockendon and A.B. Tayler. The dynamics of a current collection system for an electric locomotive. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 1971.

[105] Mustafa Ergen, Student Member, Duke Lee, Raja Sengupta, and Pravin Varaiya. WTRP — Wireless Token Ring Protocol. *IEEE Transactions on Vehicular Technology*, 2004.

[106] Sanjit Kaul, Marco Gruteser, Vinuth Rai, and John Kenney. Minimizing age of information in vehicular networks. *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON*, 2011.

[107] Sanjit Kaul, Roy Yates, and Marco Gruteser. Real-time status: How often should one update? *IEEE INFOCOM*, 2012.

[108] Clement Kam, Sastry Kompella, Gam D. Nguyen, and Anthony Ephremides. Effect of message transmission path diversity on status age. *IEEE Transactions on Information Theory*, 2016.

[109] Maice Costa, Marian Codreanu, and Anthony Ephremides. Age of information with packet management. *IEEE International Symposium on Information Theory*, 2014.

[110] Roy D. Yates. Status Updates through Networks of Parallel Servers. *IEEE International Symposium on Information Theory - Proceedings*, 2018.

[111] Ahmed M. Bedewy, Yin Sun, and Ness B. Shroff. Optimizing data freshness, throughput, and delay in multi-server information-update systems. *IEEE International Symposium on Information Theory*, 2016.

[112] Clement Kam, Sastry Kompella, Gam D. Nguyen, Jeffrey E. Wieselthier, and Anthony Ephremides. Controlling the age of information: Buffer size, deadline, and packet replacement. *IEEE Military Communications Conference MILCOM*, 2016.

[113] Clement Kam, Sastry Kompella, Gam D. Nguyen, Jeffrey E. Wieselthier, and Anthony Ephremides. Age of information with a packet deadline. *IEEE International Symposium on Information Theory*, 2016.

[114] Yoshiaki Inoue. Analysis of the Age of Information with Packet Deadline and Infinite Buffer Capacity. *IEEE International Symposium on Information Theory - Proceedings*, 2018.

[115] Roy D. Yates. Lazy is timely: Status updates by an energy harvesting source. *IEEE International Symposium on Information Theory*, 2015.

[116] Baran Tan Bacinoglu and Elif Uysal-Biyikoglu. Scheduling status updates to minimize age of information with an energy harvesting sensor. *IEEE International Symposium on Information Theory*, 2017.

[117] Xianwen Wu, Jing Yang, and Jingxian Wu. Optimal Status Update for Age of Information Minimization with an Energy Harvesting Source. *IEEE Transactions on Green Communications and Networking*, 2018.

[118] Ahmed Arafa, Jing Yang, Sennur Ulukus, and H. Vincent Poor. Age-Minimal Transmission for Energy Harvesting Sensors with Finite Batteries: Online Policies. *IEEE Transactions on Information Theory*, 2020.

[119] Haoyue Tang, Jintao Wang, Linqi Song, and Jian Song. Scheduling to Minimize Age of Information in Multi-State Time-Varying Networks with Power Constraints. *57th Allerton Conference on Communication, Control, and Computing*, 2019.

[120] Rajat Talak, Sertac Karaman, and Eytan Modiano. Improving Age of Information in Wireless Networks with Perfect Channel State Information. *IEEE/ACM Transactions on Networking*, 2020.

[121] Rajat Talak, Igor Kadota, Sertac Karaman, and Eytan Modiano. Scheduling Policies for Age Minimization in Wireless Networks with Unknown Channel State. *IEEE International Symposium on Information Theory*, 2018.

[122] Rajat Talak, Sertac Karaman, and Eytan Modiano. Distributed Scheduling Algorithms for Optimizing Information Freshness in Wireless Networks. *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, 2018.

[123] Roy D. Yates and Sanjit K. Kaul. Status updates over unreliable multiaccess channels. *IEEE International Symposium on Information Theory*, 2017.

[124] Rajat Talak, Sertac Karaman, and Eytan Modiano. Minimizing age-of-information in multi-hop wireless networks. *55th Allerton Conference on Communication, Control, and Computing*, 2018.

[125] Shahab Farazi, Andrew G. Klein, and D. Richard Brown. Fundamental bounds on the age of information in multi-hop global status update networks. *Journal of Communications and Networks*, 2019.

[126] Baturalp Buyukates, Alkan Soysal, and Sennur Ulukus. Age of information in multihop multicast networks. *Journal of Communications and Networks*, 2019.

[127] Roy D. Yates. The Age of Gossip in Networks. *IEEE International Symposium on Information Theory*, 2021.

[128] Augustin Chaintreau, Jean-Yves Le Boudec, and Nikodin Ristanovic. The age of gossip. *ACM SIGMETRICS Performance Evaluation Review*, 2009.

[129] Jori Selen, Yoni Nazarathy, Lachlan L.H. Andrew, and Hai L. Vu. The age of information in gossip networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013.

[130] John Christopher. The Asymptotic Density of Some k-Dimensional Sets. *The American Mathematical Monthly*, 1956.

Genuine Paper Versions

As is stated in the doctorate rules and regulations set by the department and the university, the papers that this dissertation is built upon have to be included in their "Originaltext" (original text / form). To comply with this rule, the section at hand contains a print of the papers in the version in which they were published.

Predictive Network Control and Throughput Sub-Optimality of MaxWeight
`https://doi.org/10.1109/EuCNC.2018.8442615`

pages 175 – 180

Model-Predictive Control for Discrete-Time Queueing Networks with Varying Topology

https://doi.org/10.1109/TCNS.2021.3074250

pages 181 – 192

Stability Results on Synchronized Queues in Discrete-Time for Arbitrary Dimension Topology

`https://doi.org/10.48550/arXiv.2006.14277`

pages 193 – 199

Distributed MPC with Prediction of Time-Varying Communication Delay
https://doi.org/10.1016/j.ifacol.2018.12.039

pages 200 – 205

A Linear Algorithm for Reliable Predictive Network Control

`https://doi.org/10.1109/GLOCOMW.2018.8644253`

pages 206 – 211

Using AoI Forecasts in Communicating and Robust Distributed Model- Predictive Control

https://doi.org/10.1109/TCNS.2021.3124260

pages 212 – 222

An Algorithm for Exact Numerical Age-of-Information Evaluation in Multi-Agent Systems

`https://doi.org/10.1109/ICC45855.2022.9838930`

pages 223 – 228

Age-of-Information in Clocked Networks

`https://doi.org/10.48550/arXiv.2207.05516`

pages 229 – 239