Fluid flow switching servers : control and observer design

DOI:
10.6100/IR777845

Document status and date:
Published: 01/01/2014

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Fluid flow switching servers

Control and Observer design

Dirk van Zwieten

This research has been supported by
the Netherlands Organization for Scientific Research
(NWO-VIDI grant 639.072.072)

A catalogue record is available from the Eindhoven University of Technology
Library ISBN: 978-90-386-3674-0

Reproduction: Universiteitsdrukkerij Technische Universiteit Eindhoven

Fluid flow switching servers

Control and Observer design

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op dinsdag 9 september 2014 om 16.00 uur

door

Dirk Antoon Jan van Zwieten

geboren te Venray

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. L.P.H. de Goey
1e promotor:	prof.dr.ir. I.J.B.F. Adan
2e promotor:	prof.dr.ir. J.E. Rooda
copromotor:	dr.ir. A.A.j. Lefeber
leden:	prof.dr.ir. S.P. Hoogendoorn (TU Delft)
	prof.dr. G. Weiss (University of Haifa)
	prof.dr.ir. A.G. de Kok

Preface

Ever since I was a child I was fascinated by how things work. This has led, mainly due to a lack of fine motor skills, to many broken toys and equipment but it improved my insights. This interest remained while growing up and here I am now, at the end of an important period in my life. By years of practice, my research skills have developed from demolition to creation (I wish I could say the same about my motor skills). Nevertheless, I am proud of what I have accomplished so far.

This thesis is a milestone in my life and the outcome of four years of research, performed within the Manufacturing Networks group of the Mechanical Engineering Department at the Eindhoven University of Technology. The comfortable research environment and colleagues made it possible for me to spend almost a decade at the University. Apart from meetings in the Benelux, I was privileged to present my work or attend workshops in Fortaleza, Lund, Munich, Milan, Phoenix, Philadelphia, Saint Petersburg and Turin, for which I am grateful to my supervisors. In the course of my research I have received help and support from many people. Here is my chance to acknowledge their contributions.

First of all I would like to thank Koos Rooda as my second promotor. He is the person that triggered me during my final master project to continue doing research at the TU/e and this has led to this PhD project. Thank you for your excellent supervision and support from the first day I joined your group. Ivo Adan, my first promotor, for his enthusiastic support and for the careful reading of my papers and this thesis. My coach Erjen Lefeber, who cooperated on most of the presented research, for his support, patience, advice, ideas and understanding.

All colleagues at the Manufacturing Networks Group and the Systems Engineering Group for the pleasant working environment they have created. Special thanks go to my (former) office mates: Allan, Damian, Konstantin, Lennart, Qin, Ricky and Stijn.

The external Doctorate Committee members Serge Hoogendoorn, Ton de Kok and Gideon Weiss whose valuable suggestions have improved the quality of this thesis. Maurice Heemels, who improved Chapter 7 with his detailed, insightful and valuable comments.

The students who I have worked with performing a final bachelor project, internship or final master project: Simon Bus, Rens van Bussel, Kees Duisters, Frans Hoogenboom, Xander Koolen, Emile Manni, Jaap Schuit, Frans Verbruggen en Jan van der Vleuten. Thank you for your contributions.

Finally, my family and friends for their support, interest and invaluable distraction. I would like to thank my parents for their endless support and Maartje for her patience and love.

Summary

In a world where the complexity of systems is ever increasing, the demand for control strategies is ubiquitous. To this end, systems of switching servers are regarded in this research. Switching servers attend multiple queues, while switching between attending queues might take time or involve costs. Systems of switching servers are all around, for instance at the manufacturing industry, food processing facilities or computer communication networks. Switching servers can also describe the dynamics of everyday life situations, such as, for example, traffic flows at signalized traffic intersections or queueing in hospital rooms. To gain more insight into the (basic) dynamics of switching servers, the switching server is modeled as a fluid flow system with constant arrival rate, constant service rate and without disturbances. It is natural to assume the flow of objects as continuous for systems where discrete objects arrive in a high volume. Systems with low-volume discrete arrivals fall outside the scope of this research, as models for discrete items are more suitable, since each individual object may have a large effect on the system.

In this research, control of switching servers is decoupled into an optimal periodic behavior problem and an optimal transient behavior problem. The optimal periodic behavior is the desired reference trajectory, which leads to optimal long-term performance of the system. The optimal transient behavior is the trajectory that optimally steers the system towards the periodic behavior, if the system is removed from the optimal periodic behavior, e.g., due to service maintenance or service priorities.

A method is presented to derive the optimal periodic behavior for a two-queue single switching server. Analytical derivation of the optimal periodic behavior for switching servers with more than two queues or with restrictions on queue contents or service periods quickly becomes too difficult, if possible at all. Therefore, for systems with setup times, setup costs and/or backlog, the optimal periodic behavior problem is formulated as a Linear Programming (LP) problem, or a Quadratic Programming (QP) problem. This method is flexible in the performance criteria and constraints used. Additional constraints can be easily implemented, e.g., bounds on queue contents or bounds on service and cycle times.

Next, this method is extended to derive the optimal periodic behavior for multi-queue single switching servers. In these systems, multiple queues can be served simultaneously at a queue-dependent rate. Hence, signalized traffic intersections, where multiple vehicle lanes can receive a green light at the same time, can also

be modeled by multi-queue switching servers. The optimization method consists of two consecutive steps. First, all feasible sequences are generated, as for systems with more than two queues, the optimal switching order is unknown a priori. Second, the optimal service times are derived for each sequence. We allow for a queue to have multiple service periods in a sequence and show, by means of examples, that allowing multiple service periods for a queue in a cycle can have a substantial (positive) effect on the system performance.

For a network of switching servers, where objects flow via predefined routes to (several) servers in the network, a similar approach is presented to derive the optimal periodic behavior. However, unlike queues for single switching servers, the arrival rate for a queue in the network can be non-constant. In the network, fluid travels between queues, which leads to piecewise constant arrival rates at queues, i.e., the piecewise constant departure rate from a queue is the arrival rate at the downstream queue. Therefore, the service periods are divided into different phases and aggregation of queues is used to derive the queue contents in the network. For the aggregation of queues, it is assumed that transportation times between queues are zero, i.e, after service the flow is immediately available in the successive queue. Then, the optimization for each feasible sequence is formulated as a QP problem and optimal periodic behavior for networks of switching servers can be determined.

Given the predetermined optimal periodic behavior, optimal transient behavior of single switching servers is investigated next. A method is presented that optimally steers the system towards the periodic behavior. For any initial state, the optimal transient trajectory problem is formulated as a QP problem. For a two-queue switching server, the system with and without backlog are considered separately. By combining the switching points, i.e., states on the optimal transient trajectory at which the server switches to the other queue, the optimal control policy can be derived for the unconstrained system without backlog. This policy is expressed in terms of switching curves: if a point on this curve is reached, the system must switch to serve the other queue. For systems with constraints on queue contents or service periods, these switching curves may not exist in general, as the switching points are affected by the constraints and they will depend on the initial state. This method can be extended to multi-queue switching servers. However, for these systems the sequences, i.e., order of serving queues, is (usually) not predefined. Feasible sequences can be generated similar to the sequence generation process for deriving the optimal periodic behavior of multi-queue switching servers. Then, for each sequence an optimization problem can be formulated and solved, and the best solution yields the optimal transient behavior.

The controllers resulting from the aforementioned approaches are central controllers and determine the switching behavior of each server, based on the state of the entire system, i.e., global state information. For non-artificial systems, e.g., signalized traffic intersections, access to global state information is hardly ever the case in practice. This renders the design of observers, providing good estimates of the global state of the system based on input/output from the system, of crucial importance. We present, as a first step towards observer design for a general switching server, observer design for multi-queue single switching servers with a clearing policy. Moreover, these systems, being highly relevant in the context of manufacturing and traffic applications, are part of a special class of piecewise affine hybrid systems. Although all subsystems are unobservable and not all events are visible, a continuous-time observer is constructed which guarantees that the estimate converges to the state of the plant under suitable conditions. The main idea is to sample the system at visible events, for which an observer can be designed using standard techniques from control theory. Then, this discrete-time observer is used as a blueprint for the continuous-time observer, where besides the plant dynamics additional 'waiting' modes are assigned to the observer. It is formally shown that these principles result in a successful observer design.

For the Kumar Seidman network, and a specific policy, a similar approach is used to design an observer, as a first attempt in the direction of observer design for networks of switched systems. Although a minimal amount of information is measured, an observer is derived which converges to the current network state. The approach is threefold. First, based on the policy and network dynamics, the switching pattern (which is cyclic) is determined. Second, the network is sampled with varying sampling periods at visible event times, and the dynamics between these times are derived. Third, the observer evaluates all possible predicted states at the visible event times. By eliminating infeasible possibilities, finally a single estimated network state remains. Via simulation it is shown that this state converges to the actual network state.

Finally, problems related to control and observer design of switching servers are discussed. For a network of two switching servers including transportation times, a heuristic is presented to derive a good service schedule for networks of switching servers. First, the network is regarded as a single server and the optimal periodic behavior for this server is determined. Second, the resulting service schedule is implemented at the network and for each server the optimal phase delay is determined. Also, scheduling of switching servers with stochastic arrival processes are briefly discussed. We approximate the average queue length for systems with stochastic arrival processes and use these approximations to derive service schedules via a non-linear programming problem. Last, optimal control policies for discrete-time controlled dynamical systems with uncertain demand is derived. Using techniques from robust optimal control, without assuming a certain strategy a priori, the policy can be derived. We also show that by the use of observers, for detectable or observable systems, unknown states can be reconstructed.

This dissertation can serve as a starting point for future research on control and observer design for switching servers. The introduced methods for determining the optimal periodic and transient behavior and the observer design can be extended to networks with stochastic behavior and can be investigated for real life control problems.

Contents

Chapter 1

Introduction

1.1 Dynamic processes

All around dynamic processes take place. Literally, dynamic stands for "*characterized by continuous change, activity, or progress*" and process for "*a series of actions, changes, or functions bringing about a result*". Then, the combination dynamic process describes a continuous change of actions that results in something. In this thesis, we restrict ourselves to controllable dynamic processes. That is, dynamic processes for which the outcome can be steered or influenced. Two examples of such dynamic processes are presented below. In Figure 1.1a, a signalized traffic intersection is depicted. The flow of vehicles crossing this intersection is a dynamic process and these vehicles are regulated by the use of traffic lights. The second example is the back-end production of a semiconductor production process, i.e., assembly and test of individual semiconductors, see Figure 1.1b. Here, the overal process is to sequentially assemble and test individual semiconductors.

(a) Signalized intersection

(b) Semiconductor production

Figure 1.1: Two examples of dynamic processes

To gain insight into dynamic processes or to control dynamic processes a *dynamic system* is developed, a mathematical description that can express and model the behavior of the dynamic process over time. A dynamic system can be classified based on the type of the state:

- *continuous state*: The state takes values in a continuous set and can take a value between any other two values. Examples of continuous states are the speed of a vehicle, the duration of a green signal at traffic intersections or the amount of fluid in a reservoir.
- *discrete state*: The state takes values in a countable or finite discrete set, such as the number of vehicles waiting in front of the traffic light or the signal of the traffic signal (either green, yellow/orange or red).

Also, a dynamic system can be categorized based on the time over which the state evolves:

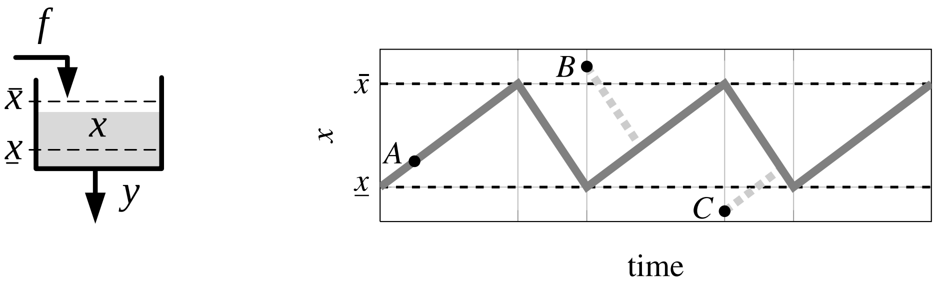
- *continuous time*: The time is a continuous set, between any two points in time there are an infinite number of other points in time. One can think of the queue length in front of a traffic signal or the traffic signal itself which both can take a value at an unspecified point in time.
- *discrete time*: The state is viewed at separate (discrete) points in time and is unchanged throughout each time period. Thus, a variable jumps from one value to another as time moves from one time period to the next. This corresponds to a digital clock that gives a fixed reading of 11:23 for a while, and then jumps to a new fixed reading of 11:24, etc. In this framework, each variable of interest is measured once at each time period. The number of measurements between any two time periods is finite.

At last, a distinction between two time-advance approaches can be made:

- *time-driven*: The state of the system changes as time advances, either continuously for continuous time systems or after every time period for discrete time systems.
- *event-driven*: The state of the system changes due to occurrence of an event, i.e., the start or end of an activity. The times between occurrences of events are typically not equidistant. Typical examples of event-driven systems are manufacturing systems, telecommunication networks and logistic systems. For a signalized traffic intersection, possible events are: a change of the traffic signal, arrival of a vehicle, or a queue becoming empty.

An example of an event-driven continuous-time discrete-state system is a vehicle queue in front of a traffic light. The state of the system is the number of vehicles waiting in the queue and the events are arrivals or departures of vehicles. Another example is the water tank depicted in Figure 1.2a, which is an event-driven continuous-time continuous-state system. Here, the state of the system is the water level x. Fluid is added via a pump with rate f to the tank and fluid is removed from the tank via an outlet with rate y.

Systems with combinations of continuous and discrete states, continuous and discrete time or time-driven and event-driven dynamics are labeled *hybrid systems*. Hybrid systems can be found in many fields and disciplines, such as traffic management systems, manufacturing systems, process control, embedded systems, mechanical and bio-mechanical systems, electrical circuits and biological systems. A simple example of a controlled hybrid system is the regulation of the water level $x(t)$ at time t in the water tank depicted in Figure 1.2a. In a simplified description, the fluid inflow rate f can be either zero or a constant value λ, i.e., mode 'off' or mode 'on' respectively. The fluid outflow rate $y < \lambda$ is constant. In each mode, the water level $x(t)$ can be described by a function based on the current water level and the in- and outflow rates. If the water level reaches the threshold $x(t) \leq \underline{x}$, the fluid inflow is switched on (mode 'on'). The inflow is switched off (mode 'off') if the water level exceeds the upper boundary $\bar{x}$.



(a) Water tank system

(b) Water level reference (solid)

Figure 1.2: Water tank system overview (left) and water level (right).

In this thesis, fluid flow switching servers are considered. These continuous-time event-driven hybrid systems have both continuous states (queue contents) and discrete states (server modes). A point of emphasis in this thesis is *control* of systems. A system can be controlled in many ways. For instance, for the water tank system depicted in Figure 1.2a, a controller regulates the inflow of water. The controller switches the system to mode 'on' or to mode 'off', depending on the water level. A typical example from control theory is a system that has to follow a *reference* and the controller manipulates the inputs of the system to follow this reference. An illustrative example is presented in Figure 1.2b. Here, the fluid level of the water tank system is depicted over time by the solid line. This trajectory is also the reference trajectory. The controller uses this reference trajectory to determine the control actions (switch the inflow 'on' or 'off'), given the state of the system, i.e., the water level $x(t)$. If the system is initially in point A in the figure, the system is located on the reference trajectory and the controller aims to follow this trajectory as good as possible. Due to disturbances, e.g., an extra inflow of water due to rain or an unpredicted outflow due to leakage, the water level can deviate from the reference trajectory, illustrated by points B and C in the figure. Then, a trajectory is derived to bring the system back to the reference trajectory, indicated by the dotted lines.

The control problems discussed in this thesis can be divided into two subproblems. The first problem is to find the *optimal periodic behavior*, that is, the periodic reference trajectory that optimizes a *performance criterion*. For signalized traffic intersections, the performance criterion can be, for example, to minimize the average queue lengths, to minimize fuel consumption or to minimize the average delay of a vehicle. Given the optimal periodic behavior, the second subproblem is to find the *optimal transient behavior*, or optimal transient trajectory. That is, steering (controlling) the system from an arbitrary starting point to the periodic behavior. Examples of transient trajectories are depicted by the dashed lines in Figure 1.2b, steering the water tank system from points B or C back to the reference trajectory.

Another point of emphasis is *observer design*. Consider again the problem of controlling the water tank system in Figure 1.2a, or any other hybrid system. For this system, the water level is required by the controller to determine a control action, i.e., switch the inflow ‘on’ or ‘off’. However, it can be that continuous water level measurements are unavailable and, for instance, only a sensor is available that indicates if the water level has reached the lower threshold $\underline{x}$. Still, after measuring the output of this sensor and the inflow and the outflow of water in the system for a while, the water level can be derived. So, to overcome the problem of missing water level measurements, an *observer* for the water level can be designed, based on both the sensor information and measurements of the inflow and the outflow of the system. The observer estimates the current water level of the system. Subsequently, the controller uses this estimate obtained from the observer instead of the actual water level measurements.

1.2 Contributions

The main contributions of the research presented in this thesis are:

- Derivation of the optimal periodic behavior for multi-queue single switching servers and for a network of switching servers. The optimal behavior is defined with respect to cycle time, time averaged weighted queue contents, time averaged weighted waiting time, queue contents or a combination of these criteria.

- Optimal control of multi-queue single switching servers. Derivation of optimal transient behavior that steers the system, with optimal performance, to the optimal periodic behavior.

- Observer design for a special class of hybrid systems, which includes multi-queue single switching servers that operate using a clearing policy. These observer design principles are also applied to design an observer for a specific network of switching servers, operated under a specific service policy.

1.3 Outline of the thesis

A graphical overview of the structure of this thesis is presented in Figure 1.3. Each box indicates a chapter. Boxes with a gray outline indicate that single switching servers are considered, black outlined boxes indicate that networks of switching servers are considered. Emphasis on control is presented in chapters with white filled boxes, emphasis on observers is presented in chapters with a grey filled box. Solid arrows indicate a strong relationship between chapters, dashed lines present a weak relationship between chapters. Note that Chapters 1 and 2 are omitted from Figure 1.3 as these chapters are introductory chapters and are linked to all other chapters.

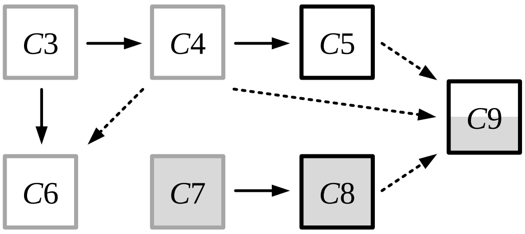


Figure 1.3: Graphical overview of the structure of this thesis.

Next to graphical thesis overview, a concise description of each chapter is given below.

Chapter 2 gives a description of fluid flow switching servers. Single switching servers and networks of switching servers are discussed. Also an overview of control and observer design of these systems is given.

Chapter 3 presents a method to derive optimal periodic behavior for a two-queue fluid flow switching server. The optimal periodic behavior is derived for systems with setup times, setup costs, backlog and constraints on queue length and service periods.

Chapter 4 presents the derivation of optimal periodic behavior for a multi-class fluid flow switching server. In the considered systems, multiple queues can be served simultaneously and a queue can be served multiple times in a cycle. A traffic application with corresponding optimal periodic service schedules is presented. In addition, the chapter contains a case study of an intersection in Eindhoven, the Netherlands, to demonstrate the proposed method.

In *Chapter 5*, the optimal periodic behavior for a network of multi-class fluid flow switching servers is presented, as an extension to Chapters 3 and 4. The considered networks are assumed to have instant transportation, i.e., zero transportation times.

Chapter 6 provides a method to derive the optimal transient behavior for a two-queue fluid flow switching server, i.e., the cost-optimal way to return to the periodic behavior from any initial state.

In *Chapter 7*, observer design for multi-class switching servers with clearing policies is presented. Based on the limited (partial) information provided by the server, it is shown that the complete state of the system can be reconstructed.

Chapter 8 presents observer design for a well known network of switching servers, the Kumar Seidman network, given a specific policy. This work is a first attempt in the direction of observer design for networks of switched systems.

Chapter 9 discusses control of a network of traffic intersections with transportation times, control of traffic intersections with stochastic arrivals and control and observer design for supply networks. these problems are related to control and observer design of fluid flow switching servers.

Finally, *Chapter 10* concludes this thesis with a summary and suggestions for further research on the subjects.

In each chapter, illustrations are provided to demonstrate the basic concept that is treated.

Chapter 2

Multi-queue switching servers

A general description of a multi-queue switching server is a server that can attend a single or multiple (but not all) queues at a time, while switching between queues might take time or involves costs. One can think of switching servers as man-made systems which can be found in, for example, the manufacturing industry, food processing facilities or computer communication networks. Switching servers can describe much more than artificial systems, as they can also be used to describe the dynamics of everyday life situations, such as, for example, traffic flows at signalized traffic intersections or queueing in hospital rooms.

In switching servers, different phenomena occur, such as maintenance, human aspects or fluctuations in arrival rate and service rate. If possible, incorporating all these phenomena in a model results in a detailed, yet complex, system. To gain more insight into the basic dynamics of the switching server, *models* are derived. Models are abstractions of detailed real systems in which only a couple of phenomena are taken into account. To this end, the switching servers are considered as *fluid flow* models, similar to the general framework introduced by [59]. Therefore, the queues in the system are represented by reservoirs that store arriving fluid and provide the fluid to the server. In lots of systems, individually distinguishable objects are directed through the system. One can think of cars at traffic intersections or wafers in the semiconductor industry. This class of systems, with discrete (countable) objects, can be regarded as *discrete event* models. Fluid flow models regard the objects that require service from a higher level of abstraction, by assuming that individual objects can not be distinguished and hence treating them as a continuous flow. For systems handling large amounts of objects, distinguishing all individual objects is not necessary, and it can even distract from the basic dynamics. Examples of such systems are, for instance, cars in a traffic intersection during rush-hour or food in the production of mass consumables.

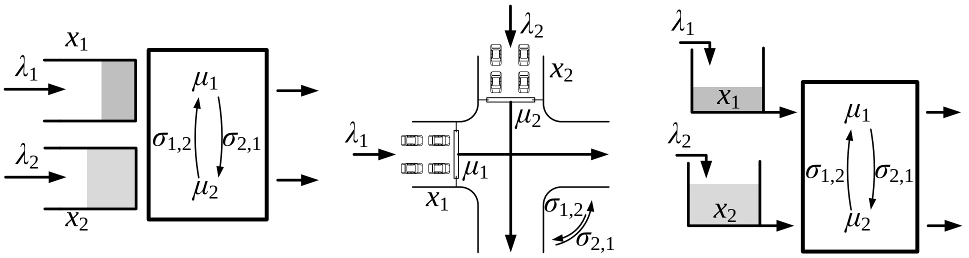
A lot of research has been conducted on the behavior of switching servers with stochastic arrival and service rates. In this thesis, the emphasis is on *deterministic* systems, i.e., constant arrival and service rates, as a better understanding of the crux of controller design for switching servers is required first. Note that, in reality, deterministic systems are rare. However, investigating deterministic systems provides insights and methods that can be relevant to controller design for stochastic systems, as can be seen in Section 9.2. Furthermore, the considered system structure does not change over time, i.e., the systems have a fixed structure, e.g., manufacturing systems or traffic networks.

The remainder of this chapter presents an overview of multi-queue switching servers and introduces the notation. In Section 2.1, a single switching server is presented. Here, the most simple system, a two-queue switching server, is introduced. Section 2.2 presents a network of switching servers. Control of switching servers is discussed in Section 2.3. Finally, observers are discussed in Section 2.4 and Section 2.5 provides an outlook for the rest of the thesis.

2.1 Single switching server

Switching servers are considered that attend multiple queues. The total number of queues in a system is denoted by $N > 1$ and the set of queues is given by $\mathcal{N} = \{1, 2, ..., N\}$. Therefore, a single switching server competes over N queues. In this section, the most simple model of a multi-queue switching server is introduced, the two-queue switching server ($N = 2$).

Several examples of the two-queue switching server are presented in Figure 2.1. These examples illustrate a manufacturing system serving two different product types (Figure 2.1a), a signalized traffic intersection with two flows of vehicles (Figure 2.1b) and a fluid flow system with two different types of fluid (Figure 2.1c). Note that all of these examples have the same behavior.



(a) Two product manufacturing system. **(b)** Two flow signalized intersection. **(c)** Two tank fluid flow system.

Figure 2.1: Different two queue switching server layouts.

For the presented systems, fluid flows representing products arrive at buffers, vehicles arrive at vehicle lanes and fluid flows arrive at tanks or reservoirs. In the remainder, we denote the location where fluid is stored, i.e., for example, buffers, lanes or reservoirs, by *queues*. Fluid arrives at each queue $i = 1, 2$ with *constant* arrival rate λ_i. The content of queue i at time t is denoted by $x_i(t)$.

The two-queue switching server is limited to serve only one queue at a time. If the server attends queue i, the service rate is given by $r_i \in [0, \mu_i]$. Typically, switching service between different queues implies a setup process. The setup process consists of *setup times* $\sigma_{i,j} \geq 0$ for switching from queue i to queue j, *setup costs* $s_{i,j} \geq 0$ or a combination of these.

When the switching server attends a queue, two successive periods can be distinguished, a *service* period and an *idle* period. A service (idle) period is defined as the uninterrupted interval during which the queue is (not) served. Note that during a setup no queue content is processed, hence a setup is part of the idle period. The duration of a service (idle) period for queue i is nonnegative and is denoted by τ_i (τ_i^0). The idle period is divided into a part during which the switching server sets up to serve the queue and a part in which the switching server idles.

Throughout the thesis, examples of signalized traffic intersections or manufacturing systems are presented. Therefore, an overview of the terminology for these systems is presented below.

Manufacturing system

A manufacturing machine working on two product types is presented in Figure 2.1a. Products (fluid) arrive at a predefined buffer (queue). The buffer content can also be denoted by the inventory level. The machine processes the products at a given process (service) rate. Once the machine requires to process the other queue, a setup time, sometimes also referred to as switchover time, might be required. During the setup, for example, the machine configurations can be adjusted or the system can be cleaned.

Traffic intersection

An isolated signalized traffic intersection can also be modeled as a single switching server. Each flow of vehicles leaves the intersection via a single route. Therefore, if vehicles that arrive from the same direction leave the intersection in distinct directions, these are modeled as multiple flows. Vehicles halt in front of a red light, before the stop line, in a vehicle lane (queue). Once a flow receives a green light, the vehicles discharge at a saturation (service) rate. Assuming constant arrival and constant saturation rates are reasonable for real world intersections. Constant arrivals can be calculated, especially in recurrent congestion, based on historical data. Also, when queues exist during green (service) periods, departure rates can be assumed constant, since vehicles discharge at their saturation rate. A green period indicates the interval during which a flow receives a green light and a red (idle) period indicates the interval at which the traffic signal is red. The amber/orange signal is not considered. This could either be modeled as part of a green signal, part of a red signal or a combination of both signals depending on the assumptions: If during the first part of an amber signal traffic still departs and during the remainder of the amber signal traffic does not depart, the first part of the amber signal can be considered as part of a green signal and the remainder as part of a red signal. Switching between flows implies a setup time, also referred to as clearance time, lost time or intergreen time. This time is reserved for vehicles to leave the intersection after the flow has received a red light, thereby preventing collisions. The setup times are based on the layout of the intersection and therefore switching to and from a flow does not imply that the same setup times are needed.

2.2 Network of switching servers

When switching servers interact, i.e., fluid served at a switching server is send to the queue of another switching server, the system is labeled as a *network* of switching servers. The most important difference with the single switching server models is that the arrival rate of fluid at queues is not always constant. Below, an illustrative example of a network of switching servers is discussed.

In a fluid flow queuing network, fluid flows from one switching server to another, and eventually leaves the network. The network consists of multi-queue switching servers, labeled by $j = 1, 2, \dots, S$. Fluid flows through the network via a single or multiple predefined routes. A route, starting from an external source, indicates which switching servers are visited and also in which order. The queues are labeled in the order of occurrence along the route(s) by $n = 1, 2, \dots, N$. Note that all N queues are divided over S switching servers. As example, a well-known manufacturing network is presented in Figure 2.3, as introduced in [62]. This manufacturing network is used as a running example throughout the thesis. The network consists of two switching servers. Server 1 serves queues 1 and 4 and server 2 serves queues 2 and 3. The network processes a single fluid flow arriving from an external source with constant rate $\lambda_1(t) = \lambda$ at queue 1. The fluid flow consecutively visits servers 1, 2, 2, and 1 via queues 1, 2, 3, and 4 respectively. Queue 1 is the only queue receiving fluid from an external source, the other queues receive the fluid from preceding queues. Switching between service of queues 1 and 4, and between service of queues 2 and 3 requires a setup period. The difference with single switching server systems is that the arrival rate of fluid in each queue is not constant. Consider, for instance, the content of queue 2 depicted in Figure 2.3. Here, the arrival rate is zero if queue 1 is not served and it is equal to the service rate of queue 1 $(r_1(t))$ otherwise.

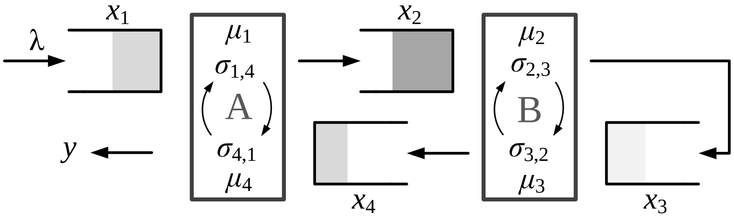


Figure 2.2: Two switching server, four queue network.

In [62] it was shown analytically by means of a counterexample that when using a clearing policy, such networks might become unstable. In a clearing policy, a queue is served until it is empty and then the server switches to attend another queue, also known as exhaustive service. Even systems with sufficient capacity at each switching server and deterministic arrival and service rates might become unstable. The main reason is because clearing policies spend too long on serving a single queue. This results in starvation of other servers and therefore capacity is wasted. Due to this waste, the effective capacity of these other servers is not sufficient anymore, resulting in an unstable system. An example of an unstable system is presented in Figure 2.3. The upper figure presents the queue contents of the queues served by server 1 (x_1 and x_4) and the lower figure presents the queue contents x_2 and x_3, served by server 2. Although the evolution of the queue contents during only 500 time units is presented, the system is unstable $\lim_{t\to\infty} x_1(t) = \infty$. Furthermore, in [62] it is also shown that even for networks without setup times, a clearing policy might result in an unstable system.

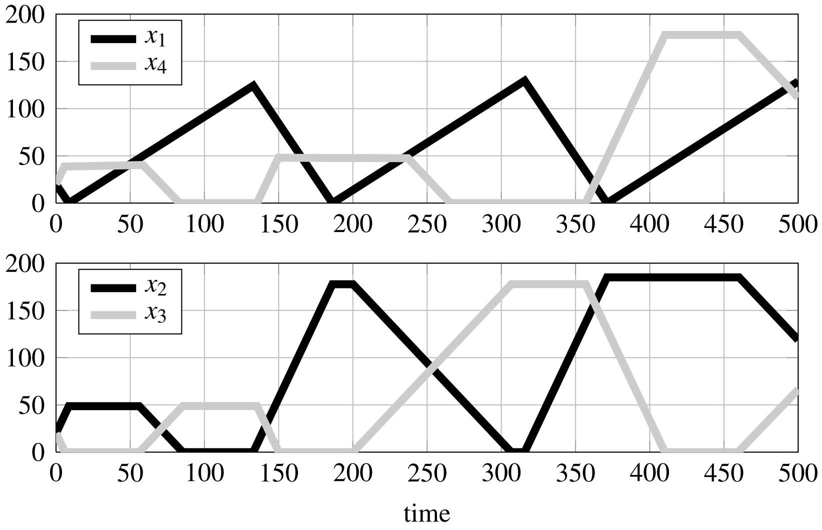


Figure 2.3: Queue content evolution, network with clearing policy.

2.3 Control

The control of hybrid systems is a problem of importance. In, e.g., [47, 72], recent overviews for various techniques for controller synthesis of hybrid systems are presented. More specifically, we focus on optimal control of switching servers. For switching servers, optimal control is relevant as, for example, optimal schedules for manufacturing systems can reduce costs via, for instance, lowering the required storage capacity and shortening lead times. For traffic signals at signalized intersections, optimal schedules can reduce congestion, and thereby reduce the amount of environmentally harmful emissions, and improve mobility.

Control of switching servers with constant arrival and constant service rates can be divided into two subproblems. The first problem is to find the *optimal periodic behavior*, i.e., a cyclic schedule of service periods or the steady-state trajectory. The second problem is to find the *optimal transient behavior*, that is, steering (controlling) the system from an arbitrary starting point (queue contents, time, mode of the server, etc.) to the periodic behavior. A parallel can be drawn with a motorist driving to a certain destination using guided navigation. The navigator computes a route, or reference trajectory, a priori. Then, the motorist uses this reference trajectory to determine the control actions (speed, orientation, etc.) given the state of the vehicle. Similarly, for a system of switching servers, first feasible (periodic) behavior should be specified. Along with the current state of the system, this behavior can be used as reference for the servers to determine what steps to take. No matter what the state of the system is, even if it is outside the periodic behavior, suitable control actions should be determined (transient behavior).

Due to the complexity of the systems, even the most simple system, a switching server attending two queues, has been investigated by many researchers. In this research, the systems incorporate different phenomena. A small overview is given here: Systems with setup times [9, 19, 33, 36, 39, 49, 53, 73, 84], setup costs [53, 73, 84], finite queues [19, 33, 36, 53, 73], backlog [19, 73] or service period constraints [44, 55]. Also, most of these authors assume symmetric systems [19, 33, 49, 53, 73]. However, a general framework that incorporates all aforementioned phenomena is lacking. In Chapter 3, we present a method to derive the optimal periodic behavior of a two-queue switching server that incorporates all phenomena, i.e., setup times, setup costs, (in)finite queues, backlog, and a heterogenous system. This behavior is optimal within the given constraints.

Determining optimal periodic behavior for a single switching server with more than two queues is an even more challenging problem, c.f. [91]. In [63], a convex optimization problem is presented for a single multi-queue switching server, which determines a, not necessarily achievable, lower bound on the waiting costs in a deterministic environment with both setup times and setup costs. Similar, polling systems (see [70, 101] and references therein) or the economic lot scheduling problem (see [82] and references therein) have been studied by a large number of researchers. For a system with a number of competing unlimited queues, negligible setup times and cost of operation per unit linear in the queue content, it is well-known that the optimal policy is a μc-rule, see [14] or [24]. This policy allocates service to the non-empty queue with the largest rate of cost decrease. In Chapter 4 we assume that the switching server is able to serve several queues *simultaneously*, unlike the above mentioned papers. Moreover, each queue is served at a queue-dependent rate, which is independent of the number of queues being served. These kind of models may arise when studying, e.g., multiclass queueing tandems, multi-server polling systems with overtaking constraints or signalized traffic intersections. These systems are also studied in [66], where optimal control of a multiclass fluid flow network for which several queues can be served simultaneously is presented. We extend this model by considering external arrival rates in each queue and non-negligible setup times.

Since the introduction of traffic lights, different models, methods and strategies for controlling isolated traffic intersections with pretimed signals, which can be regarded as multi-queue switching servers where multiple queues can be served simultaneously (see Chapter 4), have been proposed, c.f. [18, 78] and references therein. For these systems, the problem consists of determining the optimal periodic behavior or signal timing plan, i.e., scheduling the green and red periods of each flow during one cycle at the intersection. This problem is obviously important as optimal control can, for instance, improve mobility, decrease congestion and lower emissions harmful for the environment. As discussed above, analytical derivation of optimal periodic behavior is restricted to intersections with two flows. In practice, however, intersections almost always involve more than two flows. Therefore, we focus on optimal periodic behavior of an isolated intersection with multiple flows, by using optimization techniques in Chapter 4.

Research on optimizing the performance of isolated signalized traffic intersections can be roughly divided into three approaches; group-based, phase-based and lane-based.
In the group-based approach, both the composition and order of groups are *preliminarily fixed*, c.f. [4, 5, 88], and the green periods are optimized. However, by preliminarily assuming the composition of groups, one can not assure a global optimal periodic solution.
In the phase-based approach, the staging of flows is derived from knowledge of the crossing compatibility of the flows, c.f. [23, 41, 54, 90, 102, 74]. Sequences of groups, with a *single* green interval for *each flow* in a cycle, are generated. However, by means of an example, we show that giving each flow a single green interval in a cycle is not necessarily optimal and in some cases, substantial improvement is possible by allowing multiple green periods per flow.
The lane-based approach combines the *design of lane markings* and signal settings for isolated signal-controlled intersections, see for instance [97, 98] and the references therein. However, in this thesis, *fixed* lane markings (for existing intersections) are assumed. Therefore, lane-based methods do not apply in the current setting.

The control of networks of multiple switching servers is also challenging. A network can be divided into two classes: *acyclic* and *non-acyclic* networks. If switching servers are ordered such that fluid flows only towards queues of higher ordered servers, the network is acyclic. Otherwise, if such an ordering is not possible, the network is non-acyclic. Examples of non-acyclic networks are networks of signalized traffic intersections, where vehicles can both drive from one intersection to the other and vice versa, or manufacturing systems where a product route visits a specific switching server multiple times (re-entrant servers). For non-acyclic networks where each server has sufficient capacity, it is shown via simulation in [12] that queues can explode over time, i.e., the system is *unstable*. This depends on the policy used to control the system. Different clearing policies have been introduced in [80]. For deterministic single switching servers and multi-queue acyclic queueing networks with sufficient capacity, these policies are stable. However, non-acyclic deterministic networks might become unstable using a clearing policy, as shown in Section 2.2 and in [62]. This observation led to the development of queue regulators or gated policies, see [52, 79]. Here, each queue contains a gate and is therefore split into two parts. Switching to serve other queues now depends on the queues after the gate, instead of the total queue contents. Then, a server might switch earlier, avoiding long service periods on a queue. Under certain conditions these regulators stabilize the (possibly non-acyclic) network. However, note that non-acyclic networks are only unstable for certain conditions and stability can not always be easily checked. Therefore, the queue regulators are not always necessary. Fruitlessly applying these regulators results in a larger amount of fluid in the network, which is undesired for the performance. The references mentioned above constitute only a small part of the total literature in this area. However, most of them have one common approach: the system behavior is based on an a priori considered policy. An advantage of this approach is that stability is guaranteed and that the policies can be applied to any network. However, guaranteed stability does not imply good behavior. Therefore, it is better to have a general approach to design a tailor made policy for the considered system, depending on the desired performance. This is an entirely different view at the problem of controlling a network of switching servers. Instead of starting from a policy and analyzing the behavior of the system, one can start from a desired network behavior, specified a priori. For a specific well-known network presented in [62] (see Figure 2.3), this approach has been presented in [68], along with the optimal periodic behavior. In Chapter 5, we present a more general framework to derive the desired periodic network behavior for systems without transportation times. The controller which makes the network behavior converge to this desired behavior is still a topic for further research. A method can be to derive the control actions based on Lyapunov's direct method, designing a controller that continuously lowers the energy of an , 'energy-function', i.e., the error dynamics of the system, so that the system eventually settles down to the reference trajectory. This, and similar approaches are discussed in, for instance, [38, 65, 67, 103].

Another method, concerning a two-queue switching server, minimizes the total costs to converge to the desired periodic behavior from any arbitrary state, presented in [33, 45, 73]. The considered systems are either symmetric and a clearing policy is optimal, or do not include setup times. In Chapter 6, we present a method for these (heterogenous) systems, including backlog and setup times, to derive the (transient) trajectory to steer the system to the optimal periodic behavior with lowest costs. This is a transient control problem, occurring in case of a machine which is failure prone, or in case of a signalized traffic intersection which gives priority to busses, see [100]. Furthermore, control of two-queue switching servers is considered in [49, 67].

2.4 Observers

Due to numerous amount of potential applications, hybrid systems attracted a lot of attention. A great many of results on modeling, analysis, verification and control appeared in the literature. In most of these developments it is assumed that the hybrid system model is accurate and that all states, both discrete and continuous, can be measured. The latter, however, is hardly ever the case in practice. For example, in control, robotics and manufacturing, usually only a limited number of variables are measurable. This renders the *design of observers* for hybrid systems, providing good estimates of both continuous and discrete states, of crucial importance. The observer design problem is formulated as follows:

- given a hybrid system and input/output measurements, construct an observer that estimates the current state of the system as accurate as possible.

Despite this high practical relevance, there are only a few results on hybrid observer design. In [2, 3, 87], the problem of constructing observers for switching systems has been solved by assuming the perfect knowledge of the currently active mode. For the design of most observers, see, e.g. [7, 10, 11, 13, 58, 81, 99] and the references therein, the subsystems are required to be observable or detectable. For other hybrid systems, observers are designed when the switching signal is known a priori, see [93, 94]. Another approach of observer design is based on moving horizon estimation (MHE). For the state estimation problem of a dynamic system with nonlinearities and disturbances, MHE has emerged as a powerful technique, see [1, 40] and references therein. This approach is applicable to the general class of piecewise affine systems. However, implementing observers from this approach may be limited by the high computational effort.

Theoretical results regarding the fundamental question of the existence of an observer are related to notions such as final state observability, reconstructability and final state determinability on which also some work in the area of hybrid systems appeared, see, for instance, [8, 11, 15, 25, 93].

In this thesis we present observer design for a class of piecewise affine hybrid systems, of which the full details are presented in Chapter 7. The considered hybrid system is autonomous with the mode dynamics consisting of constant drift and the output within a mode being constant. Only at some times the output reveals information about the currently active mode. In particular, this implies that all subsystems are unobservable, eliminating many currently available solutions for synthesizing hybrid observers proposed in the literature. A methodology is proposed for designing continuous-time observers. This methodology consists of a few main steps. First, the system is sampled (with varying sampling periods) at so-called visible event times, i.e., times at which the output changes during a mode transition, resulting in a linear time-varying periodic system. Based on the resulting sampled system a periodic discrete-time observer is derived with the guarantee that the observer's state converges asymptotically to the (original) system's state. Next, this observer is used as a stepping stone for designing an observer in continuous time. This requires the inclusion of additional modes in the observer structure and additional reset laws at visible event times to ensure the asymptotic recovery of the system's state.

2.5 Outlook

In this chapter, an overview of multi-queue switching servers has been given. Single multi-queue switching servers and networks of switching servers are discussed. The next chapters elaborate on the behavior of these multi-queue switching servers. In Chapter 3, periodic behavior of single two-queue switching servers systems is investigated. These results are used in Chapter 4 to investigate the periodic behavior of single multi-queue switching servers, focussing on signalized traffic intersections. In these systems, the switching server can serve multiple queues simultaneously. Also a case study of an intersection in the city of Eindhoven, the Netherlands is presented. This work is extended to periodic behavior of networks of switching server in Chapter 5, where non-constant arrival rates at queues complicates the derivation. Here, the Kumar-Seidman network is used throughout the chapter as an illustrative example. Transient behavior, returning the system to the periodic behavior, is investigated for a two-queue switching server in Chapter 6. Chapters 7 and 8 investigate observer design, respectively for a single switching server and a specific network, given predefined control policies. In Chapter 9, some problems related to control and observer design of fluid flow switching servers are discussed.

Chapter 3

Periodic behavior of a two-queue server

In this chapter we study the periodic behavior of a single two-queue switching server with constant arrivals at each queue. For systems with setup times, setup costs, backlog or a combination of these, the periodic behavior is presented. To study the periodic behavior of switching servers, first the most simple system is studied, a two-queue switching server. The approach to derive the periodic behavior presented in this chapter, is used as a stepping stone for the derivation of the periodic behavior for a multi-queue switching server in Chapter 4 and for a network of switching servers in Chapter 5. Also, the periodic behavior is used as reference for the transient behavior to converge towards, presented in Chapter 6.

In most work, analysis and optimization is done within a given family of policies, such as clearing policies where a queue is served until it is empty, or threshold policies where a queue is served until a certain queue content has been reached. In the current chapter, we derive a schedule that is *periodic*, and, other than that, we do not restrict to any policy a priori. Recall that, in Section 2.1, the two-queue switching server has been introduced. In Section 3.1 a detailed system description is presented. The optimal steady-state problem is addressed in Section 3.2, along with a few performance criteria. Illustrations of periodic behavior are given in Section 3.3.

3.1 System description

We consider a system of two queues ($N = 2$) served by a single switching server, see, for instance, Figure 3.1 which depicts a two-queue switching server. Fluid arrives at each queue $n = 1,2$ with arrival rate λ_n. The content of queue n at time t is denoted by $x_n(t)$. In case of backlog, denoting an accumulation over time of work waiting to be done or orders to be fulfilled, the queue contents can be negative. Therefore, the queue contents are divided into an *inventory level* $x_n^+(t) = \max(x_n(t), 0)$ and a *backlog level* $x_n^-(t) = \min(x_n(t), 0)$. Hence, $x_n(t) = x_n^+(t) + x_n^-(t)$ and at each time instance either the backlog or inventory level is zero, i.e., $x_n^+(t)x_n^-(t) = 0$, $\forall t$. For a

This chapter is partly based on [107].

system without backlog, $x_n(t) = x_n^+(t)$, and the indicator $^+$ is often omitted. With two queues in the system, the server is limited to only serve one queue at a time. Otherwise, if both queues can be served simultaneously, no switches are required and the queues are always empty. For systems with multiple queues, a server can serve multiple queues, if allowed, as presented in Chapter 4. If the server serves queue n, the service rate is given by $r_n \in \{0, \lambda_n, \mu_n\}$. In [35] it is shown that for systems without backlog, under optimal policies, a server, attending a queue, does not idle and serves at *maximal* rate, i.e., either at rate λ_n or μ_n.

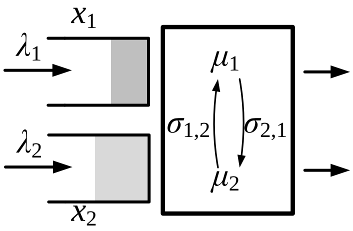


Figure 3.1: Switching server with two queues.

Typically, switching service between different queues implies a setup process, either a *setup time* $\sigma_{i,j}$ for switching from queue i to queue j, *setup costs* $s_{i,j}$ or a combination of these. A setup time can be, for instance, reserved for vehicles to leave the intersection after the queue has received a red light (end of service), thereby preventing collisions, or for a machine to adjust configurations or to do some cleaning. In the latter case, also switching costs might be involved.

When the server attends a queue, two successive periods can be distinguished, an idle period and a service period. A service (idle) period is defined as the uninterrupted interval during which the queue is (not) served. Note that during a setup no queue content is processed, hence a setup is part of the idle period. The duration of a service (idle) period for queue n is nonnegative and is denoted by τ_n (τ_n^0). The idle period is divided into a part during which the server sets up to serve the queue and a part in which the server idles. When serving a queue without backlog, it is optimal to serve the queue at the highest currently possible rate, after which the server might idle, as shown in [35]. Therefore, the service period is also divided into two parts,

$$\tau_n = \tau_n^\mu + \tau_n^\lambda,$$

where the duration of serving queue n at maximal rate is indicated by τ_n^μ and the duration of serving at arrival rate by τ_n^λ. The latter duration is referred to as *slow-mode*, since capacity is wasted. However, as indicated in [36] and shown in Section 3.3, using a slow-mode might lead to optimal behavior, since it enlarges the cycle time and thereby reduces the fraction of time spent on setups, which also wastes capacity. Also note that the order of service rates of a server that attends a queue is fixed. First the server sets up to serve the queue and possibly idles, i.e., service at rate 0.

Second, the server serves the queue at the maximal rate and finally it can serve the queue the at arrival rate.

We consider periodic behavior and a single period, or cycle, consists of successive service of both queues. That is, each queue is served once in a cycle. The *state* x of the system not only consists of queue levels x_1 and x_2, but also of the *remaining idle time* x_0 and *group* $g \in \mathscr{G}$. A group is a set of queues that can be served simultaneously and the set of all groups is denoted by $\mathscr{G}$. Given that both queues can not be served simultaneously, only two groups exist for the system considered in this chapter, denoted by $g_1 = \{1\}$ and $g_2 = \{2\}$ ($\mathscr{G} = \{\{1\},\{2\}\}$), indicating that the server sets up to or serves queue 1 and queue 2, respectively. Note that in a multi-queue system multiple queues can be served simultaneously. Therefore, not always all groups are regarded in a cycle, see Chapter 4. The remaining idle time x_0 indicates the required idle time before the start of serving a queue, and is composed of the setup time and (possible) idle time. Then, the state of the system is defined by

$$x(t) = \begin{bmatrix} x_0(t) & x_1(t) & x_2(t) & g(t) \end{bmatrix}^\top \in \mathbb{R}^3 \times \mathscr{G}. \tag{3.1}$$

The corresponding continuous dynamics are given by

$$\dot{x}_0(t) = \begin{cases} -1 & \text{if } x_0(t) > 0, \\ 0 & \text{if } x_0(t) = 0, \end{cases} \tag{3.2}$$

$$\dot{x}_n(t) = \lambda_n(t) - r_n(t), \qquad \forall n = 1,2, \tag{3.3}$$

and a switch from group i to group j causes jumps in both the state variable ($g := j$) and the remaining setup time ($x_0 := \sigma_{i,j}$), where := represents 'is set to'.

The *cycle time* T is the time it takes to serve both queues in a cycle. The cycle time consists of idle and service periods for each queue, i.e.,

$$T = \tau_1^0 + \tau_1^\mu + \tau_1^\lambda + \tau_2^0 + \tau_2^\mu + \tau_2^\lambda. \tag{3.4}$$

where the duration of the idle times includes the setup time,

$$\sigma_{j,i} \le \tau_i^0, \qquad i,j = 1,2, \quad j \neq i.$$

Define the workload of queue n by $\rho_n = \frac{\lambda_n}{\mu_n}$. An important notion for periodic behavior is *stability*. A system is called *stable* if all queue contents remain bounded. For switching servers, cf. [21, 26, 86], this definition is commonly used. To achieve a stable system, the system characteristics should meet the inflow, i.e., is it possible to process all incoming fluid? For the system in Figure 3.1, all incoming fluid can be processed if $\rho_1 + \rho_2 \le 1$. Note that the total workload for systems with setup times should be strictly less than 1. If the total workload equals 1, the server lacks capacity to serve the fluid that has arrived during setups. In a stable system the service periods must satisfy

$$\lambda_n T = \mu_n \tau_n^{\mu} + \lambda_n \tau_n^{\lambda} \qquad n = 1,2. \tag{3.5}$$

Condition (3.5) also ensures that the queue content at the start of the cycle is identical to the queue content at the end of the cycle, and therefore ensures *periodic* behavior. If (3.5) is not satisfied, the system behavior is *transient*, which is discussed in Chapter 6. Given the system description above, we present a method to derive the optimal periodic behavior in the next section.

3.2 Optimal periodic behavior

Multiple performance criteria exists for evaluating the behavior of the system. For the two-queue switching server with backlog, costs or cycle time are commonly used criteria. These criteria are discussed below. Other criteria, for the system without backlog, such as flow time or queue contents, are presented in Section 4.5. Note that in this chapter, only periodic trajectories at which each queue is served once are considered. Non-periodic trajectories or periodic trajectories in which the queues are served multiple times, are not taken into account. However, these trajectories can have lower costs, as illustrated by an example in Chapter 6 for a system with backlog.

Depending on the system under consideration, some restrictions can be imposed. As presented below, these constraints can originate from operational or safety issues. Note that these constraints are not mandatory, but can be included if required. Cycle time constraints can originate from, for instance, limiting the cycle time of a manufacturing system to the operator's available time or requiring a minimal cycle time for safety reasons in traffic intersections. Therefore, minimal and maximal cycle times, respectively $T^{\min}$ and $T^{\max}$, can be taken into account,

$$T^{\min} \leq T \leq T^{\max}. \tag{3.6a}$$

Furthermore, bounds on service periods, denoted by $\tau_n^{\min}$ and $\tau_n^{\max}$, can be required, e.g., minimal and maximal service (green) periods for traffic intersections. The service period constraints are imposed via

$$\tau_n^{\min} \leq \tau_n \leq \tau_n^{\max} \qquad n = 1,2. \tag{3.6b}$$

In addition to the constraints on cycle and service periods, the queue lengths can be bounded, e.g., finite queue capacity, and also a minimal queue level can be desired, i.e.,

$$x_n^{\min} \leq x_n(t) \leq x_n^{\max}, \qquad n = 1,2. \tag{3.6c}$$

where $x_n^{\min} \geq 0$ indicates a minimal queue capacity and $x_n^{\min} < 0$ indicates that the amount of backlog is bounded $(x_n^-(t) \geq -x_n^{\min})$. Note that for $x_n^{\min} \geq 0$ backlog in queue n is not allowed. Also, additional constraints can be imposed regarding service periods and/or queue contents.

For the system with (some of) the constraints (3.6), the performance criteria and optimal solutions are discussed below.

3.2.1 Cycle time

For the two-queue switching server, the *minimal cycle time* T^* required to serve all arrivals during a cycle, can be easily derived. Both idling of the server or wasting capacity due to service at arrival rate elongate the cycle time and are therefore not optimal, unless required to satisfy the constraints. We denote the total setup time in a cycle by $\sigma = \sigma_{1,2} + \sigma_{2,1}$. For the unconstrained system, i.e., without constraints (3.6), the minimal cycle time follows from $T = \rho_1 T + \rho_2 T + \sigma$. Required lower bounds on service periods, given by (3.6b) or by (3.6c) via

$$\tau_n \geq T - \frac{x_n^{\max} - x_n^{\min}}{\lambda_n}, \qquad n = 1, 2,$$

also affect the minimal cycle time. In the remainder of this chapter we assume that $\tau_n^{\min}$ is such that

$$\tau_n^{\min} \geq T - \frac{x_n^{\max} - x_n^{\min}}{\lambda_n}, \qquad n = 1, 2.$$

Moreover, if we also consider the lower bound on the cycle time (3.6a), the minimal cycle time follows from

$$T^* = \max\left(T^{\min}, \frac{\sigma}{1-\rho_1-\rho_2}, \frac{\sigma+\tau_1^{\min}}{1-\rho_2}, \frac{\sigma+\tau_2^{\min}}{1-\rho_1}\right). \tag{3.7}$$

Then, $\tau_1 = \rho_1 T^*$ and $\tau_2 = T^* - \sigma - \tau_1$ are, not necessarily unique, service periods of a steady-state trajectory with minimal cycle time.

Hence, the minimal cycle time is easily derived for the two-queue switching server. However, for a system with multiple queues (Chapter 4) or a network of switching servers (Chapter 5), the derivation is, if possible, much more complex. Therefore, we present another approach to derive the minimal cycle time, which is also used in the aforementioned chapters. We formulate the problem as a Linear Programming (LP) problem, given by

$$\begin{aligned}
&\min_{\tau_1^0,\tau_1^\mu,\tau_1^\lambda,\tau_2^0,\tau_2^\mu,\tau_2^\lambda} \quad T, && \\
s.t. \quad & \tau_n^{\min} \leq \tau_n \leq \tau_n^{\max}, && n = 1, 2, \\
& \tau_j^0 \geq \sigma_{i,j}, && i, j = 1, 2, \quad i \neq j,
\end{aligned}$$

$$\begin{aligned} T^{\min} &\leq T \leq T^{\max}, \\ \lambda_n T &= \mu_n \tau_n^{\mu} + \lambda_n \tau_n^{\lambda}, && n = 1,2, \\ \tau_n &= \tau_n^{\mu} + \tau_n^{\lambda}, && n = 1,2, \\ T &= \tau_1^0 + \tau_1^{\mu} + \tau_1^{\lambda} + \tau_2^0 + \tau_2^{\mu} + \tau_2^{\lambda}. \end{aligned}$$

The minimal cycle time is, next to being a performance criterion, also used for deriving the minimal time average costs, presented below.

3.2.2 Total costs

Total costs during a cycle is another performance criterion. This criterion is used, for instance, in [45]. Costs can arise during a switch from serving queue i to j, i.e., setup costs $s_{i,j}$. Also, costs can be related to the queue contents. We consider inventory costs c_n^+, which are proportional with $x_n^+(t)$, and backlog costs c_n^-, proportional with $x_n^-(t)$, which for instance arise when production is behind on the demand for the system depicted in Figure 3.1. This results in the following total costs J_c for the steady-state behavior

$$J_c = \int_0^T \left[c_1^+ x_1^+(\tau) + c_1^- x_1^-(\tau) + c_2^+ x_2^+(\tau) + c_2^- x_2^-(\tau)\right] d\tau + s_{2,1} + s_{1,2}. \qquad (3.8)$$

The trajectory minimizing J_c is the optimal steady-state behavior. Total inventory and backlog of a queue during a cycle can be derived regarding the service periods, due to the fluid flows and cyclic behavior. It can be seen in (3.8) that the setup costs s do not influence the optimal steady-state trajectory. However, these costs do play a role for the time average costs as performance indicator. Figure 3.2 presents the contents of queue n during a single cycle. All idle and service periods are indicated, together with the slope rates. We denote the minimal content of queue n by $\underline{x}_n$.

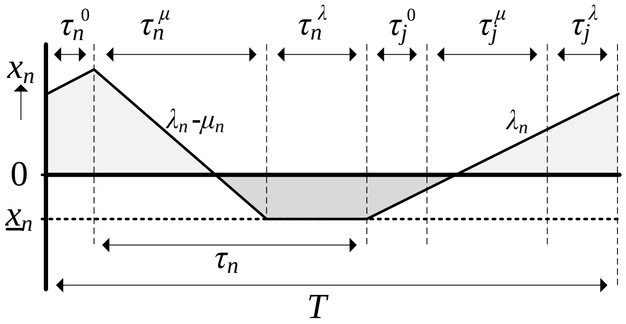
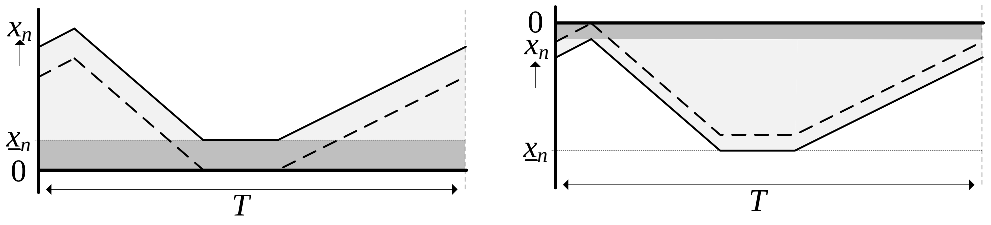


Figure 3.2: Evolution of x_n during a cycle, including setup and service periods and rates of increase/decrease.

Lemma 3.2.1. *For optimal periodic behavior it holds that*

$$\max(x_n^{\min}, (\lambda_n - \mu_n)\tau_n^{\mu}) \leq \underline{x}_n \leq \min(x_n^{\max} + (\lambda_n - \mu_n)\tau_n^{\mu}, 0), \quad n = 1,2. \qquad (3.9)$$

Proof. The proof is twofold. First, consider a trajectory where $\underline{x}_n > 0$ and $x_n^{\min} < 0$, depicted by the solid line in Figure 3.3a. Then, an alternative trajectory exists with identical service periods and $\underline{x}_n = 0$, depicted by the intersected line. The alternative trajectory has lower costs, i.e., it has $\underline{x}_n T$ less total inventory, presented by the gray area. If $x_n^{\min} > 0$, for the same reasoning, $\underline{x}_n = x_n^{\min}$ is optimal. Second, consider a trajectory where $\underline{x}_n < (\lambda_n - \mu_n)\tau_n^{\mu}$, i.e., the maximal queue content is less than zero ($x_n(t) < 0$ for all $t \in [0,T)$). This trajectory is presented by the solid line in Figure 3.3b. Then, an alternative trajectory exists with identical service periods and $\underline{x}_n = 0$, depicted by the intersected line. The alternative trajectory has lower costs, i.e., it has $(\underline{x}_n + (\mu_n - \lambda_n)\tau_n^{\mu})T$ less total backlog, presented by the gray area. If $x_n^{\max} < 0$, for the same reasoning, $\underline{x}_n = x_n^{\max} + (\mu_n - \lambda_n)\tau_n^{\mu}$ is optimal. □



(a) Trajectory with $\underline{x}_n > 0$.

(b) $\underline{x}_n < -\lambda_n(\tau_j^0 + \tau_j^{\mu} + \tau_j^{\lambda} + \tau_n^0)$.

Figure 3.3: Graphical representation of Lemma 3.2.1.

Based on Lemma 3.2.1, the inventory and backlog can be easily derived for a steady-state trajectory, presented below. Note that, for a transient trajectory, Lemma 3.2.1 does not automatically hold for each cycle. Therefore another, more complex, approach to derive the inventory and backlog levels is presented in Chapter 6. Total inventory of queue n during a cycle is denoted by w_n^+ and total backlog by w_n^-. These values depend on the queue content constraints $x_n^{\min}$ and $x_n^{\max}$, $n = 1,2$. If $x_n^{\min} \leq 0 \wedge x_n^{\max} \geq 0$, the total inventory and backlog are given by

$$w_n^+ = \int_0^T x_n^+(\tau)d\tau = \frac{1}{2}\frac{\lambda_n\mu_n}{\mu_n - \lambda_n}(T - \tau_n)^2 + \underline{x}_n T + w_n^-, \tag{3.10a}$$

$$w_n^- = \int_0^T x_n^-(\tau)d\tau = \frac{\underline{x}_n^2}{2}(\frac{1}{\mu_n - \lambda_n} + \frac{1}{\lambda_n}) + \tau_n^{\lambda}\underline{x}_n. \tag{3.10b}$$

If $x_n^{\min} > 0$, and therefore also $x_n^{\max} > 0$, it holds that

$$w_n^+ = \int_0^T x_n^+(\tau)d\tau = \frac{1}{2}\frac{\lambda_n\mu_n}{\mu_n - \lambda_n}(T - \tau_n)^2 + \underline{x}_n T, \tag{3.10c}$$

$$w_n^- = 0, \tag{3.10d}$$

and for $x_n^{\max} < 0$, the contents are given by

$$w_n^+ = 0, \tag{3.10e}$$

$$w_n^- = \int_0^T x_n^-(\tau)d\tau = \frac{1}{2}\frac{\lambda_n\mu_n}{\mu_n-\lambda_n}(T-\tau_n)^2 + \lambda_n\tau_n^\lambda(T-\tau_n) - (\underline{x}_n + (\mu_n-\lambda_n)\tau_n^\mu)T. \tag{3.10f}$$

It can be seen that the expressions for w_n^+ and w_n^- in (3.10) are quadratic in the optimization variables τ_n and $\underline{x}_n$. In case of a fixed service order and regarding the total costs, the setup costs do not influence the optimal steady-state trajectory. However, these costs do play a role for the time average costs as performance indicator.

Remark 3.2.2. *Note that the inventory of queue n during a cycle with $x_n^{\min} > 0$ can also be derived with* (3.10a) *from $x_n^{\min} = 0$, and therefore $\underline{x}_n = 0$ (Lemma 3.2.1), adding constant inventory $x_n^{\min}T$. Similarly, for $x_n^{\max} < 0$, total backlog can be derived by adding the constant backlog $-x_n^{\max}T$ to the backlog for the system with $x_n^{\max} = 0$ in* (3.10a) *(where $\underline{x}_n = x_n^{\max} - (\mu_n - \lambda_n)\tau_n^\mu$).*

For the system without backlog, the minimal total costs J_c^* can be analytically derived, which is presented in Section 3.2.3. For the system with backlog and for systems with multiple queues or networks of switching servers, discussed in the remainder of this thesis, the analytical derivation is, if possible, much more complex. Hence, we formulate the problem of finding minimal total costs as a mathematical optimization problem. Since the objective is a quadratic function of the optimization variables which are subject to linear constraints on these variables, the optimization problem is formulated as a Quadratic Programming (QP) problem, given by

$$J_c^* = s_{1,2} + s_{2,1} + \min_{\tau_n^0,\tau_n^\mu,\tau_n^\lambda,\underline{x}_n} \sum_{n=1}^{2} \left(c_n^+ w_n^+ + c_n^- w_n^-\right), \tag{3.11}$$

$$\begin{aligned}
s.t. \quad \tau_n^{\min} &\le \tau_n \le \tau_n^{\max}, && n = 1,2, && (3.12a)\\
\underline{x}_n &\le x_n^{\max} - (\mu_n-\lambda_n)\tau_n^\mu, && n = 1,2, && (3.12b)\\
\lambda_n T &= \mu_n\tau_n^\mu + \lambda_n\tau_n^\lambda, && n = 1,2, && (3.12c)\\
T &= \tau_1^0 + \tau_1^\mu + \tau_1^\lambda + \tau_2^0 + \tau_2^\mu + \tau_2^\lambda, && && (3.12d)
\end{aligned}$$

where (3.12b) follows from (3.6c), i.e., the maximal queue content is given by $\underline{x}_n + (\mu_n - \lambda_n)\tau_n^\mu$, as can be seen in Figure 3.2. Note that the objective is quadratic, as the total inventory and backlog levels (3.10) are a quadratic combination of the optimization variables.

3.2.3 Time average costs

The time average costs are commonly used as performance indicator, also referred to as time average weighted work in process for manufacturing systems or time average weighted queue lengths. In [19, 33, 37, 53, 73], this performance indicator is also considered. The time average costs, given a fixed cycle time T, are given by

$$J_w = \frac{1}{T} J_c. \tag{3.13}$$

For the system without backlog, the minimal time average costs, as well as the total costs, can be analytically derived. First, consider the unconstrained system. We assume, without loss of generality, that $c_1\lambda_1 \geq c_2\lambda_2$. Recall that, under optimal policies the server, once attending a queue, does not idle, i.e., $\tau_n^0 = \sigma_{j,n}$. The time average costs, see (3.10) for $x_n^{\min} \geq 0$, are given by

$$\begin{aligned} J_w(T) &= \frac{1}{T}\Big[a_1(T-\rho_1 T-\gamma([1-\rho_1-\rho_2]T-\sigma))^2 + \\ &\quad + a_2(T-\rho_2 T-(1-\gamma)([1-\rho_1-\rho_2]T-\sigma))^2 + s\Big] + c_n x_n^{\min}, && (3.14a) \\ a_n &= \frac{c_n}{2}\frac{\lambda_n \mu_n}{\mu_n - \lambda_n}, \qquad n = 1,2, && (3.14b) \\ \sigma &= \sigma_{1,2} + \sigma_{2,1} \geq 0, \\ s &= s_{1,2} + s_{2,1}. \end{aligned}$$

The minimal required service time is given by $(\rho_1 + \rho_2)T$. The remainder of the service time, i.e., $([1-\rho_1-\rho_2]T-\sigma)$, is divided in (3.14a) over both queues using $\gamma \in [0,1]$, which denotes the fraction of remaining service allocated at queue 1. The optimum of (3.14a) for $T > T^*$ with respect to γ is given by

$$\gamma^* = \min\left(\frac{(a_1\rho_1 + a_2\rho_1 - a_1)T + a_2\sigma}{(a_1\rho_1 + a_2\rho_1 + a_1\rho_2 - a_2 - a_1 + a_2\rho_2)T + a_1\sigma + a_2\sigma}, 1\right). \tag{3.14c}$$

Substituting (3.14b) into (3.14c) gives

$$\gamma^* = \min\left(\frac{[c_1\lambda_1(1-\rho_2) - c_2\lambda_2\rho_1]T - c_2\lambda_2\sigma}{\left[c_1\lambda_1\frac{1-\rho_2}{1-\rho_1} + c_2\lambda_2\right](1-\rho_1-\rho_2)T - c_1\lambda_1\sigma\frac{1-\rho_2}{1-\rho_1} - c_2\lambda_2\sigma}, 1\right). \tag{3.14d}$$

Note that $\gamma^* > 0$ for $T > T^*$, as the denominator of (3.14d) is strictly positive if $T > T^*$ and the nominator is zero if both $T = T^*$ and $c_1\lambda = c_2\lambda$ and strictly positive otherwise. From (3.14d), it can be found that all additional service time is allocated at serving the first queue, i.e., $\gamma^* = 1$, if

$$T \leq \frac{c_1\lambda_1\sigma}{c_2\lambda_2(1-\rho_1) - c_1\lambda_1\rho_2} \qquad \vee \qquad c_1\lambda_1\rho_2 \geq c_2\lambda_2(1-\rho_1). \tag{3.14e}$$

Hence, for $c_1\lambda_1\rho_2 \geq c_2\lambda_2(1-\rho_1)$, additional service time (slow-mode) at queue 2 is never optimal. We first present the optimization problem if (3.14e) is satisfied.

Given (3.14e), i.e., $\gamma^* = 1$, and hence all additional service is allocated at queue 1, the time average costs are given by

$$\begin{aligned} J_w(T) &= \frac{a_1(\sigma + \rho_2 T)^2 + a_2(T - \rho_2 T)^2 + s}{T} + c_n x_n^{\min}, \\ &= (a_1\rho_2^2 + a_2(1-\rho_2)^2)T + 2a_1\sigma\rho_2 + c_n x_n^{\min} + \frac{a_1\sigma^2 + s}{T}, \end{aligned} \tag{3.14f}$$

and the optimal cycle time T^{opt} can be found by minimizing (3.14f) for $T \geq T^*$, which yields

$$T^{\text{opt}} = \frac{\sqrt{\left[a_1\rho_2^2 + a_2(1-\rho_2)^2\right](s + a_1\sigma^2)}}{a_1\rho_2^2 + a_2(1-\rho_2)^2}, \tag{3.14g}$$

and $J_w^* = J_w(T^{\text{opt}})$. Note that (3.14f) is a convex function. Next, the effect of the constraints on this system are regarded. By adding bounds on the cycle time (3.6a), the feasible area changes, i.e., $T \in [T^*, T^{\max}]$. Capacity constraints, i.e., bounds on service times and bounds on queue lengths, both limit the service time duration, as discussed in Section 3.2.1. For the system satisfying (3.14e) together with minimal service period constraints, the time average costs are given by (3.14f) if $\tau_2^{\min} \leq \rho_2 T^*$, i.e., the constraint $\tau_2 \geq \tau_2^{\min}$ is inactive. Otherwise

$$J_w(T) = \frac{a_1(\sigma + \tau_2^{\min})^2 + a_2(T - \tau_2^{\min})^2 + s}{T} + c_n x_n^{\min}, \qquad \text{if } T < \frac{\tau_2^{\min}}{\rho_2}, \tag{3.14h}$$

which exceeds the time average costs given by (3.14f), as additional service time is allocated at queue 2. Note that the lower bound on the service period of queue 1 can only affect the minimal cycle time, as all extra service time is allocated at queue 1. Moreover, an upper bound on the service duration bounds the cycle time via

$$T \leq \frac{\tau_n^{\max}}{\rho_n}, \qquad n = 1, 2. \tag{3.14i}$$

If both (3.14e) and (3.14i) are satisfied, the maximal service time constraints only affect J_w if the constraint is active for queue 1, i.e, if $\tau_1^{\max} \leq T - \sigma - \rho_2 T$, as for these cycle times the duration of the slow-mode in queue 1 is limited. Then, the time averaged costs are given by

$$J_w(T) = \frac{a_1(T - \tau_1^{\max})^2 + a_2(\sigma + \tau_1^{\max})^2 + s}{T} + c_n x_n^{\min}, \qquad \text{if } T > \frac{\tau_1^{\max} + \sigma}{1 - \rho_2}, \tag{3.14j}$$

which also exceeds the time averaged costs for the unconstrained system for similar reasoning. Then, the optimal time averaged costs J_w^* for the system satisfying (3.14e) is the minimum of the following optimization problems

minimum of (3.14h) for $\qquad \max\left(T^*, T^{\min}\right) \leq T < \dfrac{\tau_2^{\min}}{\rho_2},$

$$\begin{aligned}
&\text{minimum of (3.14f) for} \quad \max\left(T^*, T^{\min}, \frac{\tau_2^{\min}}{\rho_2}\right) \leq T \leq \quad \min\left(T^{\max}, \frac{\tau_1^{\max}-\sigma}{1-\rho_2}\right),\\
&\text{minimum of (3.14j) for} \quad \frac{\tau_1^{\max}-\sigma}{1-\rho_2} < T \leq \quad T^{\max},
\end{aligned}$$

which can be easily derived. As an example, consider a heterogenous (non-symmetric) system without backlog, with parameters

$$\begin{aligned}
\lambda_1 &= 2, & \lambda_2 &= 1,\\
\mu_1 &= 8, & \mu_2 &= 4,\\
\sigma_{2,1} &= 3, & \sigma_{1,2} &= 7, \qquad (3.15)\\
c_1^+ &= 8, & c_2^+ &= 1,\\
T^{\min} &= 10, & T^{\max} &= 50.
\end{aligned}$$

A graphical representation of $J_w(T)$ for the system with parameters (3.15) (satisfying (3.14e)) is presented in Figure 3.4. The solid line presents the time average costs for the unconstrained system. The costs for the system with $\tau_2^{\min} = 10$ is depicted by the dashed line and the costs for the system with $\tau_1^{\max} = 40$ is depicted by the dotted line. Note that the optimum for the unconstrained system and system with the maximal service period constraint on queue 1 is located at $T^{\text{opt}} = 32$. If the minimal service period for queue 2 is required, the optimum is located at $T = 40$.

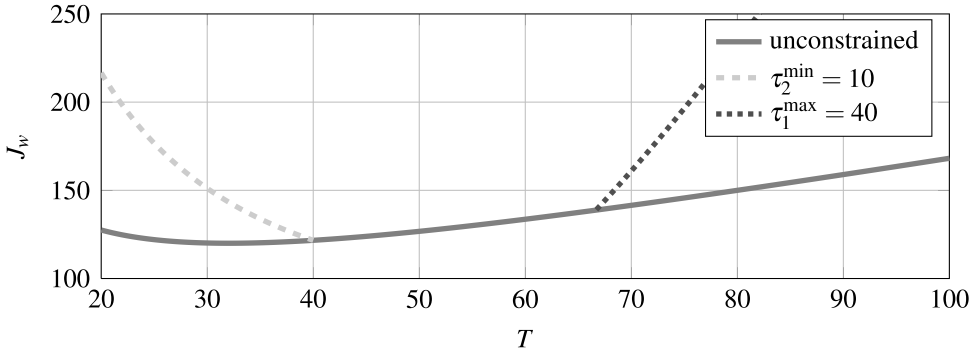


Figure 3.4: Time average costs for the system with parameters (3.15).

If the additional service time is allocated at service of both queues, i.e., (3.14e) does not hold, the time average costs are given by substituting (3.14d) into (3.14a), which yields

$$J_w(T) = \frac{c_1\lambda_1 c_2\lambda_2 (T+\sigma)^2}{2[c_1\lambda_1(1-\rho_2)+c_2\lambda_2(1-\rho_1)]T} + \frac{s}{T} + c_n x_n^{\min}. \tag{3.16a}$$

Furthermore, for the system with minimal and maximal service time constraints, the costs are given by

$$J_w(T) = \frac{a_1(T-\tau_1^{\max})^2 + a_2(\sigma+\tau_1^{\max})^2 + s}{T} + c_n x_n^{\min}, \qquad \text{if } T > \frac{\tau_1^{\max}+\gamma^*\sigma}{\rho_1+\gamma^*(1-\rho_1-\rho_2)}, \tag{3.16b}$$

$$J_w(T) = \frac{a_1(\sigma+\tau_2^{\max})^2 + a_2(T-\tau_2^{\max})^2 + s}{T} + c_n x_n^{\min}, \qquad \text{if } T > \frac{\tau_2^{\max} + (1-\gamma^*)\sigma}{\rho_2 + (1-\gamma^*)(1-\rho_1-\rho_2)}, \tag{3.16c}$$

$$J_w(T) = \frac{a_1(T-\tau_1^{\min})^2 + a_2(\sigma+\tau_1^{\min})^2 + s}{T} + c_n x_n^{\min}, \qquad \text{if } T < \frac{\tau_1^{\min} + \gamma^*\sigma}{\rho_1 + \gamma^*(1-\rho_1-\rho_2)}, \tag{3.16d}$$

$$J_w(T) = \frac{a_1(\sigma+\tau_2^{\min})^2 + a_2(T-\tau_2^{\min})^2 + s}{T} + c_n x_n^{\min}, \qquad \text{if } T < \frac{\tau_2^{\min} + (1-\gamma^*)\sigma}{\rho_2 + (1-\gamma^*)(1-\rho_1-\rho_2)}, \tag{3.16e}$$

Then, similar as presented above, the optimum is given by the minimum of all optimization results within the feasible cycle time range. If the objective is to optimize the total costs, a similar approach can be used where $J_c = TJ_w$.

To derive the optimal periodic behavior using QP, e.g., for the system with backlog or for systems with multiple queues, the cycle time T is required to be a constant value. Otherwise, as the cycle time depends on service and idle periods, the objective function (3.13) is non-linear. Then, the solution $J_w(T)$ with minimal costs for cycle times within the range

$$T^* \leq T \leq \min\left(T^{\max}, \frac{\tau_n^{\max}}{\rho_n}\right), \quad n = 1, 2,$$

renders the optimal steady-state costs J_w^*, which can be easily found.

The optimal trajectory, within the constraints, is a trade-off between loss of capacity due to setups or slow-modes and the average setup costs. Elongating the cycle time, by including a slow-mode or creating backlog, results in less switches over time where capacity is lost due to setups and thereby lowers the average setup costs. For a system without backlog, a general steady-state trajectory is depicted in Figure 3.5. The trajectory consists of six characteristic points, labeled in alphabetical order $A-F$. The optimal trajectory for systems *without constraints* contains at most one slow-mode, i.e., $F = A$ or $C = D$. Furthermore, if no setup periods are considered, $D = E$ and $A = B$. Optimal policies consist of serving queue i until the other queue j reaches a threshold. Therefore, the trajectory can include slow-modes at both queues. A special case of this model, with $\mu_1 = \mu_2$ and $c_1 = c_2$ has been studied in [19, 33, 49, 53, 73] and it is shown that the optimal policy is a *clearing* policy, i.e., the server empties a queue and then switches to serve the other queue.

3.3 Illustrations

Using the method described above, we illustrate some optimal steady-state trajectories for the two-queue switching server. We consider again the system with parameters (3.15). Note that, at first, backlog is not allowed for this system. From (3.7), we find the minimal cycle time $T^* = 20$. The corresponding periodic behavior with

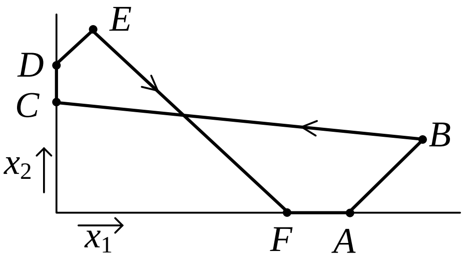
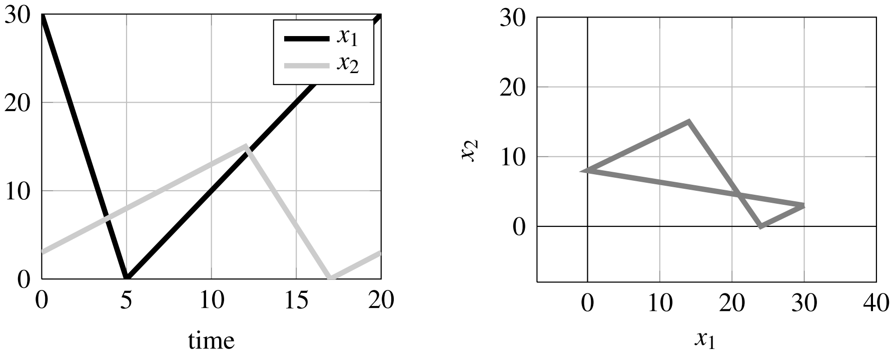


Figure 3.5: Optimal steady-state trajectory, with characteristic points $A - F$.

minimal costs is presented in Figure 3.6. Here, in the figure on the left, the evolution of the queue contents during a cycle are presented. In the figure on the right, the periodic trajectory, i.e., x_1 versus x_2, is presented. It can be seen that no slow-mode occurs in the trajectory, as expected by regarding the minimal cycle time. This trajectory also yields the optimal total costs $J_c^* = 2550$ (and $J_w = 127.5$).

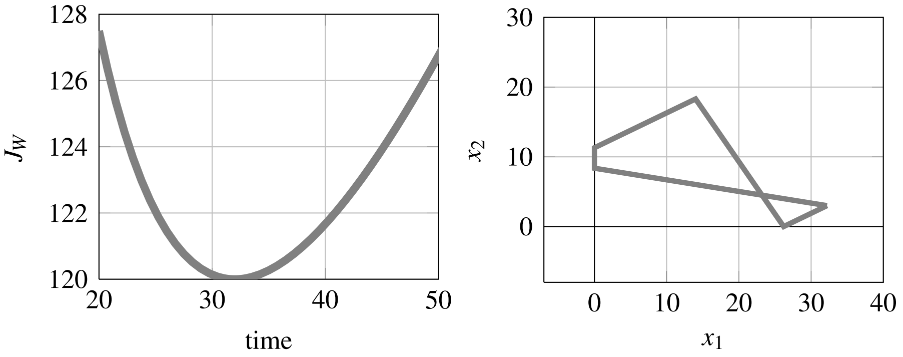


(a) Queue contents versus time. **(b)** Optimal trajectory.

Figure 3.6: Optimal queue contents over time (left) and periodic trajectory (right) during a cycle for the system with parameters (3.15) and $T = T^* = 20$.

For the time average costs as performance indicator, the trajectory with minimal cycle time does not result in the optimal trajectory. The steady-state costs versus the (feasible) range of cycle times is depicted in Figure 3.7a. The minimum is located at $T = 32$, and the corresponding trajectory, depicted in Figure 3.7b, is the optimal steady-state trajectory. The optimal costs are $J_w^* = 120$ with service periods $\tau_1^\mu = 6$, $\tau_1^\lambda = 8$, $\tau_2^\mu = 6$ and $\tau_2^\lambda = 0$.

Next, the effects of setup costs, backlog and the constraints are presented in a stepwise manner. Starting from the system with parameters (3.15), without setup costs, without constraints on service periods and no backlog, we add parameters and restrictions step by step and analyze the periodic behavior optimizing the time averaged costs. Note that the previous constraints are preserved. Adding setup costs $s_{2,1} = 300$ and $s_{1,2} = 200$ to the system results in the optimal trajectory depicted in Figure 3.8a. The optimal cycle time, costs, service periods and minimal queue contents for each trajectory are presented in Table 3.1. Compared to the trajectory depicted in 3.7b, it can be seen that the addition of setup costs elongates the cycle time and increases the duration of the slow-mode. This is also expected, it is bene-



(a) Time average costs. **(b)** Optimal trajectory.

Figure 3.7: Time average costs versus cycle time (left) and optimal trajectory for $T = 32$ (right) for the system with parameters (3.15).

ficial to enlarge the cycle time as the costs of switching increase.

Allowing backlog, with backlog costs $c_1^- = 50$ and $c_2^- = 3$, shifts the optimal trajectory downwards and enlarges the cycle time, see Figure 3.8b. For queue 2, the inventory and backlog costs are equal. Note that no backlog occurs at queue 1, which is optimal due to the long slow-mode and the high costs of backlog. Next, the service period of queue 1 is restricted ($\tau_1^{\max} = 15$), resulting in the optimal trajectory depicted in Figure 3.8c. The service period of queue 1 for this trajectory is the maximal service period, see Table 3.1. Adding upper bounds on the queue contents, $x_1^{\max} = 35$ and $x_2^{\max} = 16$, also reduces the cycle time, see Figure 3.8d. Also, both upper bounds are reached in the trajectory. In Figure 3.8e, the optimal trajectory of the system with a maximal cycle time $T^{\max} = 25$ is depicted. This trajectory has a cycle time of 20 time units, the minimal required cycle time (no slow-modes), and also queue 1 has backlog.

For systems that require a minimal amount of products in the queue, or a maximal allowed amount of backlog, we add minimal queue constraints to the model. In Figure 3.8f, the optimal trajectory is depicted where the content of queue 1 is at least 2 and the backlog of queue 2 can not exceed 2, i.e., $x_1^{\min} = 2$ and $x_2^{\min} = -2$. Unlike the previous trajectory, the optimal trajectory is not the trajectory with the minimal cycle time and queue 2 reaches both boundaries. Finally, in Figure 3.8g the optimal steady-state trajectory for this system without setup periods is depicted. Due to the setup costs, this trajectory is not the fixed point $(2,0)$. The optimal cycle time is 20 time units, which is not the minimal cycle time for this system.

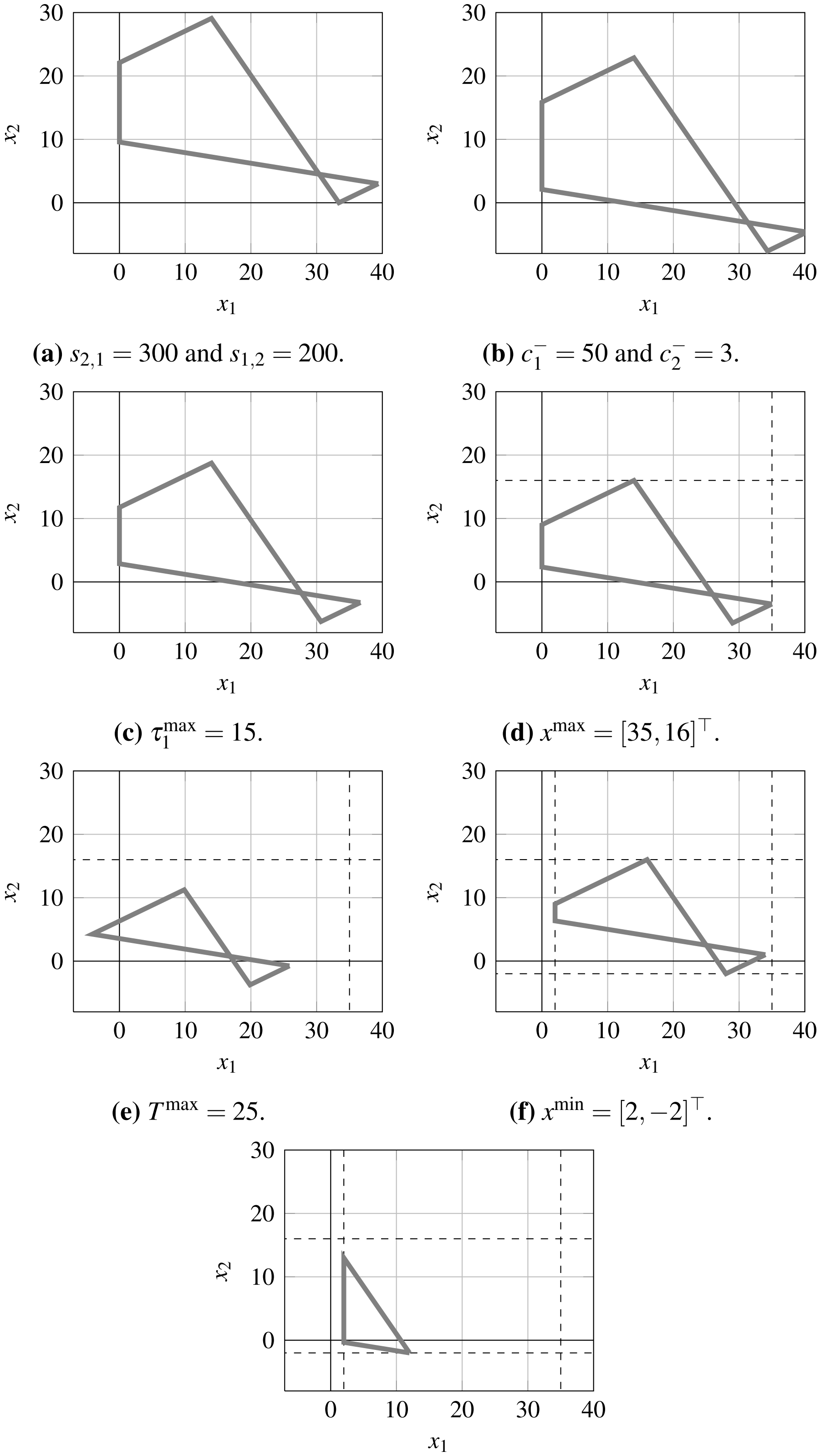
Fig.	T	J_w^*	τ_1^μ	τ_1^λ	τ_2^μ	τ_2^λ	$\underline{x}_1$	$\underline{x}_2$
3.7b	32	120	6	8	8	0	0	0
3.8a	38,78	134,13	6,57	12,52	9,70	0	0	0
3.8b	40,65	130,41	6,72	13,77	10,16	0	0	-7,62
3.8c	33,33	131,93	6,11	8,89	8,33	0	0	-6,25
3.8d	30	134,06	5,83	6,67	7,5	0	0	-6,5
3.8e	20	134,07	5	0	5	0	-4,14	-3,75
3.8f	24	158,06	5,33	2,67	6	0	2	-2
3.8g	20	60,37	1,67	13,33	5	0	2	-2

Table 3.1: Optimal cycle times, costs, service periods and minimal queue contents for the trajectories depicted in Figures 3.7-9.5.

3.4 Summary

In this chapter, the periodic behavior for a two-queue switching server has been investigated. Optimal periodic behavior, where each queue is served once in a cycle, is derived using LP or QP, depending on the performance criteria. The problems regarding cycle time, total costs or time average costs as criterion are presented. For the time average costs, the optimization problem is formulated as a QP, given a fixed cycle time. Then, the problem is solved over a range of cycle times, and the best solution renders the optimal periodic behavior. The effect of setup costs, backlog, bounds on cycle times, bounds on service periods and bounds on queue levels on the periodic behavior is presented via illustrations.

In Chapter 4, we elaborate on these results and investigate the periodic behavior for multi-queue switching servers. Here, the order of serving queues is taken into account and it is possible for a server to serve multiple queues simultaneously. In addition, in Chapter 5, the periodic behavior of a network of switching servers is investigated. The steady-state trajectories derived in this chapter are also used in Chapter 6 as reference for the transient behavior problem, i.e., returning the system to the periodic behavior.



(a) $s_{2,1} = 300$ and $s_{1,2} = 200$.

(b) $c_1^- = 50$ and $c_2^- = 3$.

(c) $\tau_1^{\max} = 15$.

(d) $x^{\max} = [35, 16]^\top$.

(e) $T^{\max} = 25$.

(f) $x^{\min} = [2, -2]^\top$.

(g) No setup periods.

Figure 3.8: Optimal steady-state trajectories for the system with parameters (3.15), minimizing the time average costs. The system is subsequently extended with setup costs (a), backlog (b), maximal service period (c), maximal queue contents (d), maximal cycle time (e) and minimal queue contents (f). In (g), no setup periods are considered.

Chapter 4

Periodic behavior of a multi-queue switching server

Two-queue switching servers have been studied in Chapter 3. Periodical behavior of systems with setup periods, setup costs, backlog and bounds on cycle times, service periods and queue lengths has been presented. Based on the performance criterion, the optimal periodic behavior is derived using LP or QP techniques.
This chapter proceeds with the results in Chapter 3 and studies a single switching server that serves multiple $(N > 2)$ queues. If the server can serve more than two queues, the complexity of the optimal behavior increases drastically. For the two-queue server, the order in which the queues are served is fixed, i.e., serve queue 1, queue 2, queue 1, queue 2, etc. For the three-queue server, the order in which the queues are served is not given a priori. If, for instance, the server has finished serving queue 1, it can serve either queue 2 or queue 3. Also, serving a queue more than once in a cycle can be a good option, especially when the load of the queues is unbalanced.

Moreover, the server considered in this chapter can serve multiple queues simultaneously. Each queue is served at a queue-dependent rate, which is independent of the number of queues being served. These kind of models may arise when studying, e.g., multi-class tandem queues, multi-server polling systems with overtaking constraints and signalized traffic intersections. In addition to the system studied in [66], where optimal control of a multiclass fluid flow network for which several queues can be served simultaneously is presented, we include external arrival rates in each queue and consider non-negligible setup periods.

The following class of queueing networks can be represented by a multi-queue single switching server. If, in a queueing network, fluid arrives at a constant rate at all queues, and switching between service of queues might take time, the network can be represented by a multi-queue switching server that can serve multiple queues simultaneously. Examples of such networks are presented in Figure 4.1. The network

This chapter is partly based on [104, 108].

on the left is a 4-queue two-server polling system with physical constraints such that the servers can not serve consecutive queues and the servers can not overtake each other, e.g., think of quay cranes at a container terminal serving vessels. In the middle, a traffic intersection with four flows is presented, for which conflicting flows can not be served simultaneously. The network on the right presents a four queue tandem network where each of the three servers can serve two queues and only one queue can be served at a time. Each of these networks can be modeled as a single switching server with four queues that can serve several queues simultaneously, with a fixed service rate for each queue, i.e., capacity is not shared over multiple queues.

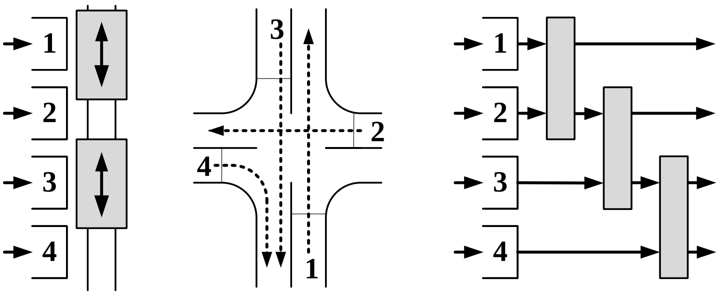


Figure 4.1: Three examples of multiclass queueing networks which can be represented by a multi-queue switching server: Two-server polling network (left), traffic intersection (middle) and multiclass queueing tandem network (right).

In the remainder of this chapter, for ease of presentation, we focus on signalized traffic intersections. Note that the presentation can also be in a general context, see Section 2.1 for more details on the translation between traffic intersections and a general system. By considering traffic intersections, backlog in the system is omitted, as the content of vehicle lanes are non-negative. Backlog can be included, with a similar, but tedious, approach as presented in Section 3.2.

The remainder of this chapter is organized as follows: An introduction to the scheduling problem of signalized traffic intersections, along with a small description of the approach is presented in Section 4.1. Section 4.2 introduces the model and problem, along with an illustrative example that is used throughout the chapter. Feasible sequence generation, step 1, is addressed in Section 4.4. The performance criteria and optimization of a sequence, step 2, is presented in Section 4.5. Section 4.6 presents illustrations of a five-flow traffic intersection and a 3-queue switching server. A real-life example is presented in Section 4.7 and a summary is provided in Section 4.8.

4.1 Signalized traffic intersections

The control problem for signalized traffic intersections consists of determining the optimal periodic behavior or signal timing plan, i.e., scheduling the green and red periods of each flow during one cycle at the intersection. This problem is obviously important as optimal control can, for instance, improve mobility, decrease congestion and lower emissions harmful for the environment. In this chapter we decompose the pretimed traffic control problem into two steps that can be considered consecutively. First, all feasible sequences are generated. Second, for each sequence the optimal green periods are derived.

The first step, generating all feasible sequences, consists of three subproblems. First subproblem is *staging of flows*, i.e., which flows receive a green light, sometimes also referred to as right of way, simultaneously? For complex intersections involving a large number of flows, the specification of the number and composition of groups is a nontrivial task, that can have a major impact on intersection capacity and efficiency. Unlike [54, 90, 102, 74] we take *all possible* groups into account, i.e., not only the maximal groups, but also all subgroups of these groups. In [41, 56] also subgroups are taken into account, though the resulting sequences are restricted to a *single* green period for each flow only.
The second subproblem consists of *combining* the groups. Which groups can be combined such that all flows receive a green period at least once? For computational reasons, we restrict ourselves to a reasonable maximal number of groups per combination of groups. However, under this restriction, we evaluate all possible combinations of groups. Hence, also combinations are considered with the same flows appearing in multiple groups. Moreover, the same group can appear multiple times in a combination.
For each combination of groups, the groups can be *ordered* in multiple ways, resulting in different sequences (or cycles) with possibly different performance. This renders the third subproblem. In a sequence it is possible that some flows are served more often than others, i.e., *multiple* green periods are allowed for some flows in a cycle. This has, to the best of our knowledge, not been addressed before and results show that this can have significant impact on performance.

In the second step, the optimal periodic behavior (or cycle) is derived for each sequence. By considering a fixed cycle time, the objective function is linear or quadratic (depending on the criteria) with respect to the green periods and constraints are linear. Hence, we use linear or quadratic programming techniques to optimize the performance, and repeat this over a range of cycle times. A variety of performance indicators or criteria exists, see for instance [32]. We consider minimizing the weighted average amount at vehicles on the intersection, weighted average waiting time at the intersection, maximal queue length, cycle time or a linear combination of these. Furthermore, unlike for instance [46], we derive a schedule that is *periodic*, and, other than that, we do not restrict to any policy a priori.

Computation of all sequences, as required in the above mentioned first step, is expensive. Instead of an exhaustive approach we use a *recursive* approach that eliminates unfeasible sequences as soon as possible by employing the constraints. To illustrate that this method is computationally affordable for real-life intersections we present the derivation of the optimal periodic behavior of the largest real-life intersection in Eindhoven, the Netherlands.

The contribution of this chapter is two-fold. First, we present a method to generate all feasible sequences of groups, including sequences with multiple green periods for flows, and which is still computationally affordable for large intersections. Second, we formulate the timing optimization of a sequence as a linear or quadratic programming problem. This formulation allows considerable flexibility in the constraints and performance criteria.

4.2 System description

A traffic intersection is modeled as a multi-queue single switching server that serves a number of queues in a fixed cyclic order. Each *flow* of vehicles $n = \{1,2,...,N\}$ corresponds to a queue at the intersection. The set of all flows is denoted by $\mathscr{N}$. The queue content of flow n at time t is denoted by $x_n(t)$, which is the number (or amount) of vehicles behind the stop line at the intersection, and is bounded from above:

$$x_n(t) \leq x_n^{\max}, \qquad n = 1,2,...,N. \tag{4.1}$$

If flow n receives a red light, the queue content x_n increases linearly with rate λ_n. While receiving a green light, the queue content depletes with rate $\mu_n - \lambda_n$, i.e., service at rate μ_n, if non-empty ($x_n > 0$) and remains empty otherwise ($x_n = 0$). Note that in the latter case, vehicles leave the intersection at the arrival rate λ_n (slow-mode).

Switching from a flow i to a flow j and $i \neq j$ implies a *setup period*, denoted by $\sigma_{i,j} \geq 0$, also referred to as clearance time, lost time or intergreen time. This period is reserved for vehicles to leave the intersection after the flow has received a red light, preventing collisions. Note that $\sigma_{i,j}$ and $\sigma_{j,i}$ need not be identical: Switching to and from a flow does not imply that the same setup periods are needed, e.g., due to the layout of the intersection. Furthermore, switching between non-conflicting flows does not require a setup period. An illustrative example of a 5-flow traffic intersection is presented in Figure 4.2. The intersection consists of three vehicle flows (flows 1, 3 and 5) and two pedestrian/bicycle flows (flows 2 and 4).

4.2.1 Service schedule

The traffic lights on the intersection operate according to a service schedule. In a schedule, the flows are divided into groups. A *group* g is a set of flows, that can receive a green light simultaneously. For instance, flows 1 and 5 or flows 2 and 4 for the intersection depicted in Figure 4.2 do not conflict and these flows can therefore receive green lights simultaneously. Corresponding groups are $\{1,5\}$ or $\{2,4\}$, respectively. The set of all groups is denoted by $\mathscr{G}$. The setup period to

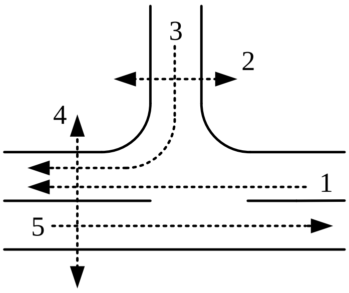


Figure 4.2: Five-flow traffic intersection.

switch from a group $g \in \mathscr{G}$ to a flow n is the *maximal* setup period between all these flows, denoted by:

$$\sigma_{g,n} = \max_{i \in g} \sigma_{i,n}.$$

A green (red) period is defined as the uninterrupted interval during which the signal is green (red), i.e., the interval between the moment that the signal is switched to green (red) and the moment that the signal is switched to red (green). The duration of a green (red) period for flow n in group $g_i \in s$ is nonnegative and is labeled green (red) period. The duration of a green period is denoted by $\tau_{i,n}$. The amber/orange signal is not considered. This could either be modeled as a part of a green signal, a part of a red signal or a combination of both signals depending on the assumptions: If during the first part of an amber signal traffic still departs and during the remainder of the amber signal traffic does not depart, the first part of the amber signal can be considered as part of a green signal and the remainder as part of a red signal. A green period starts after a red period, it can be spread over *multiple* consecutive groups and it ends before another red period.

A *sequence* s consists of an ordered list of multiple groups, e.g., (g_1, g_2, g_3) where $g_i \in \mathscr{G}$, and it is a single repetition of the service operations at the network, sometimes referred to as signal or service cycle. The groups g_i are numbered in the order they occur in the sequence, i.e., g_2 refers to the second group in a sequence. Note that we focus on periodic schedules. The time it takes to serve all groups in a sequence is called the *cycle time*, denoted by T.

Determining an optimal sequence might be too difficult, if this sequence exists at all, i.e., enlarging the sequences might result in ever increasing performance for particular situations. Therefore, we restrict the length and composition of the sequences. Within these restrictions the optimal periodic behavior is determined. First, for operational and computational reasons, the maximal number of groups in a sequence s has an upper bound given by

$$|s| \leq G. \tag{4.2}$$

Second, as a flow can receive a green signal in multiple groups and can have *multiple green periods* within a single sequence, let Θ denote the *maximal number of green* *periods* received by a specific flow during a cycle. Note that the maximal number of green periods a flow can receive, which are interrupted by red periods, is at most half of the number of groups. Receiving a green light at the first and last group of a period also counts as a single green period, as the sequence is repeated. Hence,

$$\Theta \leq \left\lfloor \frac{G}{2} \right\rfloor, \tag{4.3}$$

where $\lfloor r \rfloor$ denotes the largest integer smaller than $r \in \mathbb{R}$. Moreover, the maximal number of green periods that a flow receives during one cycle for sequence s is denoted by θ_s, therefore

$$\theta_s \leq \Theta. \tag{4.4}$$

A sequence is labeled *feasible* if all queues are served at least once and satisfies the constraints (4.2) and (4.4). The set of feasible sequences is denoted by $\mathscr{S}$. Stability of a sequence is addressed in Section 4.3. For example, a feasible sequence for the intersection depicted in Figure 4.2, also used throughout the chapter, is given by

$$\hat{s} = (\{1,2,5\},\{1,5\},\{2,4\},\{1,5\},\{3,5\}).$$

Note that sequence $\hat{s}$ consists of five groups ($|\hat{s}| = 5$, where $|\hat{s}|$ denotes the sequence length), some of which identical ($g_2 = g_4 = \{1,5\}$). Also, some queues are served more than others.

For operational purposes, it is desired to keep Θ low, since one can imagine that drivers could get irritated when another queue is served multiple times. However, allowing multiple green periods in a sequence, i.e., $\Theta > 1$, can result in much better performance, as shown via examples in Sections 4.6–4.7. For sequence s, we define the set of numbers of the groups in which flow n is served by

$$\mathscr{P}_{s,n} = \{i \in \{1,2,...,|s|\} | n \in g_i\}.$$

Let us consider sequence $\hat{s}$ for the system depicted in Figure 4.2. Then, for example, $\mathscr{P}_{\hat{s},1} = \{1,2,4\}$ and $\mathscr{P}_{\hat{s},2} = \{3\}$.

Furthermore, we define α_i as the total service period required by group g_i, i.e., time to set up and serve all queues contained in group g_i. Note that the service period of a group includes a service period of a flow that belongs to that group and the corresponding setup period, i.e.,

$$\alpha_i = \sigma_{g_{i-1},n} + \tau_{i,n} \qquad \forall n \in g_i,$$

where g_{i-1} denotes the group before g_i, i.e., $g_{i-1} = g_{1+(i-2) \bmod |s|}$. An example of a service schedule for sequence $\hat{s}$ is presented in Figure 4.3. For all flows, the red and green periods are depicted during a single cycle, represented by the dark gray and light gray areas, respectively. It can be seen that queue 5 receives a green signal in four consecutive groups (4-5-1-2), hence no setups are required for this flow at the start of the first, second and fifth group. Also, queues 1 and 2 receive two green

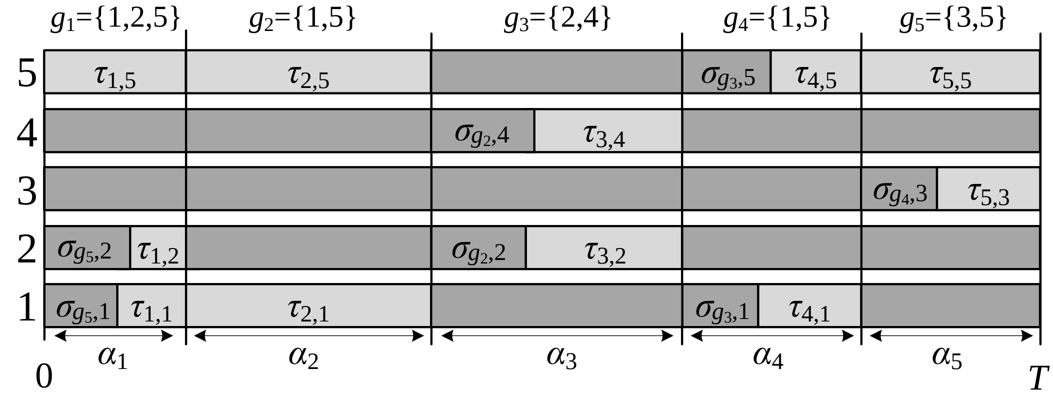


Figure 4.3: Service schedule example for sequence $\hat{s}$ for the system in Figure 4.2.

periods during a cycle.
Note that the start of a green period for a flow in a group depends on the duration of the setup period. Therefore, the starting times of green signals of the flows in a group can be different. However, the green periods do end at the same time, for all flows in the group, provided that the flow is not contained in the successive group, e.g., at the end of group $g_3 = \{2,4\}$ in the example.

For operational or safety purposes, minimal or maximal duration of green periods can be required, denoted by $\tau_n^{\min}$ and $\tau_n^{\max}$, respectively. As the green period of a flow is not necessarily restricted to a single group, we define by $P_{s,n}$ the set of maximal sets of consecutive groups in which flow n receives a green signal in sequence s. Formally, for a sequence $s = \left(g_1, g_2, ..., g_{|s|}\right)$, we define

$$P_{s,n} = \{\emptyset \neq g \subseteq \mathscr{P}_{s,n} | \exists! i \in g : 1 + (i) \bmod |s| \notin \mathscr{P}_{s,n} \wedge \\ \wedge \exists! j \in g : 1 + (j-2) \bmod |s| \notin \mathscr{P}_{s,n}\}.$$

Note that the maximal number of green periods θ_s that a flow receives during one cycle for sequence s can be derived from $P_{s,n}$, i.e.,

$$\theta_s = \max_{n=1,2,...,N} |P_{s,n}|.$$

The green period constraints for sequence s are imposed via

$$\tau_n^{\min} \leq \sum_{i \in p} \tau_{i,n} \leq \tau_n^{\max} \qquad \forall p \in P_{s,n}, \qquad n = 1, 2, ..., N. \tag{4.5}$$

For example, consider flow 1 in sequence $\hat{s}$ for the example depicted in Figure 4.2:

$$\mathscr{P}_{\hat{s},1} = \{1,2,4\}, \\ P_{\hat{s},1} = \{\{1,2\},\{4\}\},$$

which yields the following green period restrictions:

$$\tau_1^{\min} \leq \tau_{1,1} + \tau_{2,1} \leq \tau_1^{\max},$$

$$\tau_1^{\min} \leq \tau_{4,1} \qquad \leq \tau_1^{\max}.$$

Also for operational or safety reasons, the cycle time can be restricted, similar to the two-queue system presented in Chapter 3. For instance, minimal and maximal cycle times, respectively $T^{\min}$ and $T^{\max}$, have to be taken into account

$$T^{\min} \leq T \leq T^{\max}. \tag{4.6}$$

4.3 Stability

An important aspect for control of systems is stability. Given arrival and departure rates, is it possible for the intersection to serve all incoming vehicles? The system is called *stable* if there exists a periodic schedule for which all incoming vehicles during a cycle can be served, i.e., no accumulation of fluid in queues over multiple cycles. For multi-queue switching servers, where a server can serve only *one* queue at a time ($|g_i| = 1$, $i = 1, 2, ..., |s|$), stability is ensured if

$$\sum_{n=1}^{N} \rho_n < 1, \tag{4.7}$$

with load $\rho_n = \frac{\lambda_n}{\mu_n}$. A system with two vehicle flows, or two queues as discussed in Chapter 3, is stable if the total workload $\rho_1 + \rho_2 < 1$. This is obvious, as both flows can not receive a green period simultaneously. Note that the total workload is strictly smaller than 1, since this results in additional capacity to serve vehicles that have arrived during the setup periods. Note that condition (4.7) is *sequence independent*, i.e., if this condition is satisfied there exists a sequence (and green periods) for which the system is stable.

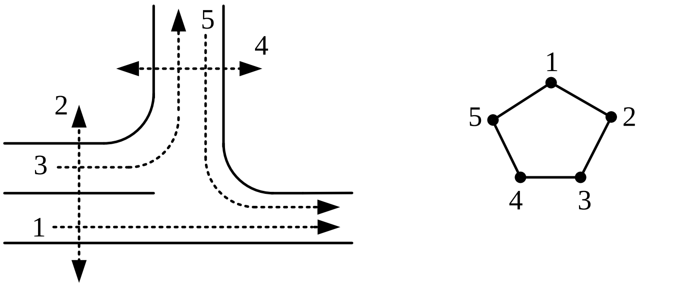
The traffic intersections considered in this chapter are able to give a green signal to multiple flows simultaneously, i.e., $|g_i| \geq 1$, $i = 1, 2, ..., |s|$, rendering stability condition (4.7) invalid. In [20], a stability condition is proven for a combination of groups that allows green signals to multiple flows simultaneously, given that each flow receives a single green signal during a cycle ($\Theta = 1$). This condition is given by

$$\sum_{i=1}^{|s|} \max_{n \in g_i} \rho_n < 1, \qquad \Theta = 1, \tag{4.8}$$

and states that the sum of the maximal ρ's of a group should be less than one, which is quite intuitive. This condition is *sequence dependent*, i.e., it tells whether a specific combination of groups, and therefore a specific set of sequences, can be stable or not. To find a stable sequence, it might be required to check whether there is a combination of groups for which condition (4.16) holds for *all* possible combinations of groups.

A method to find the stability criteria for a traffic intersection that is sequence independent is presented below. For an isolated intersection, criteria are derived that ensure existence of a stable schedule. These criteria are necessary and sufficient in the absence of constraints on cycle and green periods, respectively (4.6) and (4.5). Including these constraints renders the criteria necessary.

The flows and conflicts at an intersection can be represented by an undirected graph $\Gamma = (\mathcal{N}, \mathcal{E})$, consisting of a set of vertices $\mathcal{N}$ together with a set of edges $\mathcal{E}$. The vertices represent the flows and the edges indicate conflicting flows, i.e., if flows i and j conflict and can not receive a green signal simultaneously, there exists an edge (i, j) or (j, i) between vertices i and j in the graph. For illustrating the stability conditions, an intersection is considered with three vehicle flows and two pedestrian flows, see Figure 4.4a. This specific intersection is presented, as it is the smallest example which requires, next to the usual stability conditions, an extra stability condition, which is discussed below. The corresponding undirected graph of the intersection is presented in Figure 4.4b, with $\mathcal{N} = \{1,2,3,4,5\}$ and $\mathcal{E} = \{(1,2),(2,3),(3,4),(4,5),(5,1)\}$.



(a) Intersection layout. **(b)** Intersection graph.

Figure 4.4: Layout (left) and graph (right) of an intersection with three vehicle lanes (1,3 and 5) and two pedestrian lanes (2 and 4).

Before the stability conditions are derived, let us first define *allowed* groups:

Definition 4.3.1. *A group $g \subseteq \mathcal{N}$ is called an allowed group when g is an independent set for graph Γ, i.e., for every two vertices in g there is no edge connecting the two.*

Recall that $\mathcal{G}$ denotes the set of allowed groups, which is thus the same as the set of all independent sets of graph Γ. For the graph in Figure 4.4b, we have

$$\mathcal{G} = \{\{1,3\},\{1,4\},\{2,4\},\{2,5\},\{3,5\},\{1\},\{2\},\{3\},\{4\},\{5\},\emptyset\}. \tag{4.9}$$

For a set Q of groups, a maximal group is defined as follows:

Definition 4.3.2. *A group of Q is a* maximal *group of Q if it is not properly contained in another group of Q.*

The set of maximal groups of set Q is denoted by $Q^{\max}$. The maximal allowed groups for example (4.9) are given by

$$\mathscr{G}^{\max} = \{\{1,3\},\{1,4\},\{2,4\},\{2,5\},\{3,5\}\}.$$

Definition 4.3.3. *A* clique *is a graph in which every vertex is connected to every other vertex in the graph. Let $\mathscr{C}$ denote the set of all cliques in graph Γ.*

A clique of graph Γ indicates conflicting flows of the intersection depicted in Figure 4.4a. For the graph in Figure 4.4b the set of all cliques is given by

$$\mathscr{C} = \{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{1,5\},\{1\},\{2\},\{3\},\{4\},\{5\},\emptyset\}. \tag{4.10}$$

Clearly the sum of all ρ_i's in a clique should be less than one, as these flows can not receive a green signal simultaneously. This is a necessary condition for stability characterized by the graph Γ, given by

$$\sum_{i \in C} \rho_i < 1, \qquad \forall C \in \mathscr{C}^{\max}. \tag{4.11}$$

Only the set of maximal cliques are considered, since the conditions for cliques that are not maximal are redundant, e.g., for cliques $\{i,j\}$ and $\{i\}$ it holds that $\rho_i + \rho_j < 1$ and $\rho_i < 1$, where the latter condition is incorporated in the first condition and can therefore be omitted. Condition (4.11) for the system in Figure 4.4a results in

$$\rho_1 + \rho_2 < 1, \tag{4.12a}$$
$$\rho_3 + \rho_4 < 1, \tag{4.12b}$$
$$\rho_1 + \rho_5 < 1, \tag{4.12c}$$
$$\rho_2 + \rho_3 < 1, \tag{4.12d}$$
$$\rho_4 + \rho_5 < 1. \tag{4.12e}$$

However, condition (4.11) is not (always) sufficient. For stability of the system in Figure 4.4a, in addition to the conditions (4.12a)-(4.12e), it is also required that

$$\frac{1}{2}(\rho_1 + \rho_2 + \rho_3 + \rho_4 + \rho_5) < 1. \tag{4.12f}$$

This last requirement, which is not intuitively derived, follows from the stability problem discussed below. Without loss of generality, we assume that $T = 1$. Then, the fraction of time spent on serving group g_i during a cycle is given by α_i. For ease of reading, we abuse the notation a little bit, i.e., $\alpha_{\{i,j\}} = \alpha_k$ with $g_k = \{i,j\}$. For stability, the load of flow i can not exceed the total fraction of time spent serving flow i. For example,

$$\rho_1 \leq \alpha_{\{1,3\}} + \alpha_{\{1,4\}} + \alpha_{\{1\}} \tag{4.13}$$

is required for stability of flow 1 for the system in Figure 4.4a. The fraction of time spent serving queue 1 only, i.e., the third term on the right hand side of (4.11), can, without loss of performance, be added to the fractions of time spent serving queue 1 in the maximal groups, i.e., the first and second term on the right hand side of (4.13). This holds for all non-maximal groups. Therefore, only maximal groups are considered below. The stability problem can be written as a multi-parametric linear problem, i.e,

$$\min_{\alpha} \quad \alpha_{\{1,3\}} + \alpha_{\{1,4\}} + \alpha_{\{2,4\}} + \alpha_{\{2,5\}} + \alpha_{\{3,5\}},$$

$$\begin{aligned}
s.t. \qquad & \rho_1 \leq \alpha_{\{1,3\}} + \alpha_{\{1,4\}}, \\
& \rho_2 \leq \alpha_{\{2,4\}} + \alpha_{\{2,5\}}, \\
& \rho_3 \leq \alpha_{\{1,3\}} + \alpha_{\{3,5\}}, \\
& \rho_4 \leq \alpha_{\{1,4\}} + \alpha_{\{2,4\}}, \\
& \rho_5 \leq \alpha_{\{2,5\}} + \alpha_{\{3,5\}}, \\
& 0 \leq \alpha_i, && i \in \mathscr{G}^{\max}, \\
& 0 \leq \rho_n \leq 1, && n = 1,2,...,5.
\end{aligned}$$

Solving this problem results in different regions, each with corresponding costs (as function of ρ_n). For the setup periods also a fraction of time is required. Therefore the sum of fractions of time spent serving a group (the objective function) is strictly less than 1, i.e., $\sum \alpha < 1$. Then, the costs for each region from the optimization problem represents the stability conditions (4.12). Therefore, if the stability conditions (4.12) are satisfied, there exists a schedule for a stable (unconstrained) system. For instance, for $\rho_i = 0.4 - \varepsilon$, the only sequences that result in a stable system consists of all maximal allowed groups, e.g., $s = (\{1,3\},\{1,4\},\{2,4\},\{2,5\},\{3,5\})$. An example of a traffic intersection for which no stable schedules exist is for the system in Figure 4.4a with parameters $\rho_1 = \rho_3 = 0.9$ and $\rho_2 = \rho_4 = \rho_5 = 0.08$, corresponding to a lot of traffic on the one way street and a small amount of pedestrians and traffic from the perpendicular road. This gives

$$\rho_1 + \rho_2 = 0.98 < 1, \tag{4.14a}$$
$$\rho_2 + \rho_3 = 0.98 < 1, \tag{4.14b}$$
$$\rho_3 + \rho_4 = 0.98 < 1, \tag{4.14c}$$
$$\rho_4 + \rho_5 = 0.16 < 1, \tag{4.14d}$$
$$\rho_5 + \rho_1 = 0.98 < 1, \tag{4.14e}$$
$$\rho_1 + \rho_2 + \rho_3 + \rho_4 + \rho_5 = 2.04 > 2, \tag{4.14f}$$

and therefore no stable solution exists. In general, the stability problem is given by

$$\min_{\alpha} \quad \sum_{i \in \mathscr{G}^{\max}} \alpha_i,$$

$$s.t. \qquad \rho_i \leq \sum_{i \in \mathscr{G}^{\max}} \alpha_i \mathbb{1}_{g_i}(n), \qquad n = 1,2,...,N$$

$$\begin{aligned} &0 \leq \alpha_i, && \forall i \in \mathscr{G}^{\max}, \\ &0 \leq \rho_i \leq 1, && i = 1,2,...,N. \end{aligned}$$

with indicator function

$$\mathbb{1}_{g_i} = \begin{cases} 1 & \text{if } n \in g_i, \\ 0 & \text{if } n \notin g_i. \end{cases}$$

The system presented in Figure 4.4b is the smallest example in which extra stability conditions arise, i.e., in addition to the conditions given by (4.11). For the intersection with constraints on service or cycle time, the conditions (4.11) are necessary, but not sufficient. Therefore, additional sequence-dependent stability conditions are provided, given by

$$\rho_n T \leq \sum_{i \in g} \tau_{i,n}, \qquad \forall g \in \mathscr{P}_{s,n}, \quad n = 1,2,...,N, \tag{4.15}$$

which states that all arriving fluid can be served within a cycle. Note that the cycle time is a linear combination of service periods and setup periods.

This completes the formulation of the model and the schedule. Below we formulate the problem of determining the optimal periodic schedule.

Problem formulation

For an isolated intersection, consisting of N flows and given arrival rates, saturation rates and setup periods, derive the *optimal periodic schedule* satisfying constraints (4.2) and (4.4). In other words: Find the sequence s and periodic pretimed green periods $\tau_{i,n}$, $\forall n \in g_i, \forall g_i \in s$, satisfying the constraints, that minimizes a performance criterion.

This problem is solved in two steps. First, all feasible sequences are generated, see Section 4.4. Second, for each feasible sequence, the optimal green periods are derived, presented in Section 4.5.

4.4 Sequence generation

Feasible sequence generation is divided into three subproblems, that can be considered consecutively. These subproblems are the grouping of flows into groups, grouping of groups and deriving all feasible sequences from these groups of groups.

4.4.1 Group generation

The group generation problem consists of finding all possible combinations of flows, which can receive right of way simultaneously. This equals to finding all independent sets in a graph. The undirected graph for the illustrative example is presented

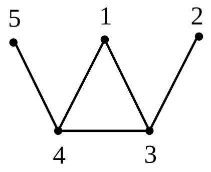


Figure 4.5: Five-flow traffic intersection undirected graph.

in Figure 4.5. Generating all groups amounts to finding all allowed groups, which is performed recursively. Allowed groups need not necessarily be *maximal* groups. The set $\mathscr{G}$ does not only include maximal groups, but also all subsets of the maximal groups. It might seem counterintuitive to take subsets of maximal groups into account, but due to flow-dependent setup periods, this may yield better performance, as shown by means of an example in Section 4.6. The set of allowed groups for the example in depicted in Figure 4.2 is given by

$$\mathscr{G} = \{\{1,2,5\},\{1,2\},\{1,5\},\{2,4\},\{2,5\},\{3,5\},\{1\},\{2\},\{3\},\{4\},\{5\},\emptyset\},$$

where $\mathscr{G}^{\max} = \{\{1,2,5\},\{2,4\},\{3,5\}\}$.

4.4.2 Combining groups

The second subproblem towards generating sequences, is to derive all feasible combinations of groups $g \in \mathscr{G}$. A combination of groups is feasible if all flows receive at least one green period and the number of groups does not exceed G. The combinations of groups are generated using a similar approach as in Section 4.4.1, by recursively enumerating all possible combinations of groups, subject to the constraints mentioned above. Next, combinations where each flow receives a single green period are checked for stability, i.e., a schedule is stable if all arriving vehicles can be served, see [20]:

$$\sum_{i=1}^{|s|} \max_{n \in g_i} \rho_n < 1. \tag{4.16}$$

This condition states that the sum of loads of the flows with maximal loads in each group should be less than one, which is quite intuitive. This is a necessary and sufficient condition in the absence of constraints on cycle and green periods. The unstable combinations of groups can be removed. Note that, in addition, a lower bound on the number of groups in a sequence can be imposed if desired. Stability for a sequence including all constraints is ensured by constraint (4.15).

4.4.3 Feasible sequence generation

Last, for each combination of groups, the groups can be ordered in different ways resulting in different sequences $s \in \mathscr{S}$. If a combination of groups consists of i groups, $(i-1)!$ different sequences arise, since w.l.o.g. it can be assumed that the first group is fixed in the sequence.

From all generated sequences, we discard the sequences that are infeasible. Sequences for which flows receive $> \Theta$ green periods are discarded. Also, sequences with two or more identical groups in consecutive order are discarded, as these identical consecutive groups can be lumped together resulting in a sequence which is already considered or not allowed. The resulting set of sequences is the set of feasible sequences $\mathscr{S}$.

4.5 Sequence optimization

With all feasible sequences derived in Section 4.4, we now present the derivation of optimal green periods for each $s \in \mathscr{S}$. From these results, the best solution renders the optimal periodic behavior. An individual sequence is optimized using linear programming (LP) or quadratic programming (QP), depending on the performance criteria.

The performance criteria are discussed first. Second, the objective functions are presented. Third, the linear constraints are addressed, which are similar for both LP and QP. Last, the generic optimization problem is presented.

4.5.1 Performance indicators

For each sequence, the optimal duration of green periods depends on the performance criteria. There exist a variety of performance criteria, relevant to the setting of traffic signals. We consider minimizing the weighted average amount of vehicles at the intersection, weighted average waiting time at the intersection, maximal queue content, cycle time, or a linear combination of these, which are discussed below.

Weighted average amount of vehicles

The weighted average amount of vehicles, or queue content, at the intersection is given by:

$$J_w = \sum_{n=1}^{N} c_n W_n, \tag{4.17}$$

with c_n the weight factor and W_n the time-average queue content of flow n:

$$W_n = \frac{1}{T} \int_0^T x_n(\tau) d\tau. \tag{4.18}$$

For ease of reading we denote $W = W^+$, as systems with backlog are not considered. The average queue content of a single flow can be described by the duration of the green and red periods during a cycle, as discussed in Section 4.5.2. The performance criterion (4.17) is used for the example in Section 4.6 and case study in Section 4.7.

Weighted average waiting time

A second criterion is the weighted average waiting time, also known as delay or flow time. The waiting time denotes the time from arriving at a queue until the vehicle has left the intersection after passing the green signal. From Little's law, see [69, 89, 48] and references therein, the following relation holds between the average waiting time φ_n and the average queue content W_n:

$$\varphi_n = \frac{W_n}{\lambda_n}.$$

Therefore, weighted average waiting time J_φ of the intersection is given by:

$$J_\varphi = \sum_{n=1}^{N} \frac{c_n W_n}{\lambda_n}. \tag{4.19}$$

So, in fact, J_w and J_φ can both be considered as a weighted average queue content. Note that the choice of weight factor c_n is important, e.g., for $c_n = 1$, $n = 1, 2, \dots, N$, J_φ represents the sum of the waiting times per queue and for $c_n = \frac{\lambda_n}{\sum_{i=1}^{N} \lambda_i}$ the objective J_φ represents the average delay per vehicle.

Maximal queue content

The maximal queue content, denoted by χ, is another criterion that can be easily incorporated in our framework, by adding constraint:

$$x_n \leq \chi, \qquad n = 1, 2, \dots, N. \tag{4.20}$$

The objective is to find the minimal χ for which (4.20) is valid.

Cycle time

Another important performance criterion concerns the minimal cycle time required to handle the existing traffic flow pattern. Moreover, we use the minimal cycle time to reduce the calculation time of optimizing the weighted average amount of vehicles and waiting time, as shown in Section 4.5.2. The minimal cycle time does not necessarily imply that the intersection is working at full capacity for the queues with the highest load in each group. Due to both constraints (4.6) and (4.5), the system might still have additional capacity at the minimal cycle time, as is shown via example in Section 4.6, which is required to handle fluctuations in arrival rates, that typically occur in practice. We return to this issue later, in Remark 4.7.1. Then, the cycle time for sequence s can be described by:

$$T = \sum_{i=1}^{|s|} \alpha_i. \tag{4.21}$$

Linear combination of criteria

Instead of optimizing a single performance criterion, they can also be linearly combined. Consider, for instance, minimizing maximal queue content. If there exists a solution, there exist infinitely many solutions governing the same result, due to combinations of processing at maximal and arrival rate. Combining this criterion with another criterion, for instance minimal average queue content, results in a unique solution. By adding weights to the criteria, emphasis can be put on a specific criterion.

4.5.2 Objective functions

The problem is to find the green and red periods for each flow for a given sequence such that overall performance is optimal. The red periods of flow n are a combination of setup periods and green periods of flows in groups, which do not contain this flow. In [35] it is shown that for optimal policies, a flow that receives a green signal is served at highest possible rate and does not idle, i.e., served at rate 0. Therefore, a green period $\tau_{i,n}$ of flow n in group g_i can be split into a duration $\tau^{\mu}_{i,n}$ where the vehicles are leaving the intersection at saturation flow rate, i.e., when queue content $x_n > 0$, and a duration $\tau^{\lambda}_{i,n}$ where the vehicles leave the intersection at arrival rate, i.e., when $x_n = 0$. Note that a green period is not bounded by a single group, e.g., if a flow n is contained in successive groups g_1 and g_2, the green period of flow n covers both groups and is denoted by $\tau_{1,n} + \tau_{2,n}$.

Using these green periods as optimization variables, the objective functions can be addressed per performance criterion. The cycle time (4.21) is the sum of green and red periods of a flow. A red period is a combination of required setup periods and green periods of other flows. Hence, the cycle time is a linear combination of several green periods. The constraints on the green periods, presented in Section 4.5.3, are linear. Note that, since the queue contents $x_n(t)$ can be expressed as linear combination of the green periods, constraint (4.20) for the maximal queue content χ is also linear. Hence, we can conclude that minimizing the cycle time and minimizing maximal queue content result in LP problems.

The average queue content W_n for $n = 1,2,...,N$, given by (4.18), is used to determine the weighted average amount of vehicles (4.17) and weighted average waiting time (4.19).
By means of an example, we illustrate the derivation of the average queue content in terms of $\tau^{\mu}_{i,n}$ and $\tau^{\lambda}_{i,n}$, $i = 1,2,...,|s|$. Consider sequence $\hat{s}$ for the system depicted in Figure 4.2, consisting of five groups and flow 1 that receives a green period twice during a sequence, e.g., $\mathscr{P}_{\hat{s},1} = \{1,2,4\}$. Note that flow 1 receives two separate green periods although it is contained in three groups. The queue content during a single cycle is presented in Figure 4.6. We denote the content of queue n at the start of group g_i by $x_{i,n}$. For ease of exposition, the subscript $_{,1}$ (denoting flow 1) is omitted in this example and we denote by β_i the red period duration before giving

flow 1 a green signal in group g_i, which is a linear combination of service periods of groups and setup periods. Note that if a flow is served multiple times in a sequence, the queue is not necessarily emptied each time being served. The total queue content during this cycle equals the area beneath the graph.
Calculation of the total queue content can be split into $|\mathscr{P}_{s,n}|$ parts, three for this example, where each part consists of a possible red period and successive green period.

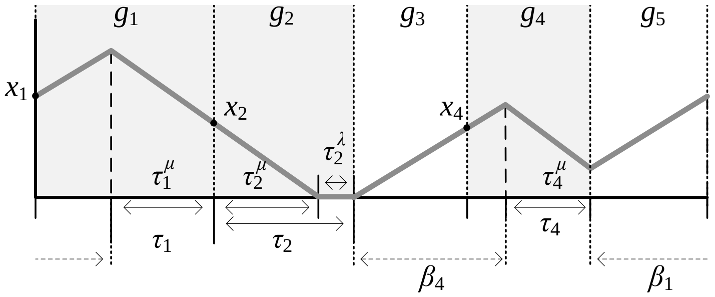


Figure 4.6: Queue content evolution of flow 1 in sequence $\hat{s}$ (served twice in a cycle).

Let us consider the first part of queue content x_1 in Figure 4.6, i.e., the evolution during $\beta_1+\tau_1$. The area can be easily computed, since this part consists of a linear increase with rate λ_n during β_1, decrease with rate $\lambda_n-\mu_n$ during τ_1^μ and constant level during τ_1^λ. Therefore, the time average queue content during $\beta_1+\tau_1$ can be described by

$$W_1 = \frac{1}{T}[(x_1+\lambda_1\sigma_{g_5,1})(\tau_1+\beta_1)-(\mu_n-\lambda_n)\left(\tfrac{1}{2}\tau_1^\mu+\tau_1^\lambda\right)\tau_1^\mu-\tfrac{1}{2}\lambda_n\beta_1^2],$$

with

$$x_1=(\beta_1-\sigma_{g_5,1}+\sigma_{g_3,1})\lambda_n+x_4-\tau_4^\mu(\mu_n-\lambda_n),$$

and, in general, we have

$$\begin{aligned} W_n &= \frac{1}{T}\sum_{i\in\mathscr{P}_{s,n}}(x_{i,n}+\lambda_n\sigma_{g_{i-1},n})(\tau_{i,n}+\beta_{i,n})-(\mu_n-\lambda_n)\left(\tfrac{1}{2}\tau_{i,n}^\mu+\tau_{i,n}^\lambda\right)\tau_{i,n}^\mu-\tfrac{1}{2}\lambda_n\beta_{i,n}^2, \\ x_{i,n} &= \lambda_n\left(T-\tau_{i,n}^\mu-\sigma_{g_{i-1},n}-\sum_{j\in\mathscr{P}_{s,n}}\tau_{j,n}^\lambda\right)-\mu_n\left(\sum_{j\in\mathscr{P}_{s,n}\backslash i}\tau_{j,n}^\mu\right), \qquad \forall i\in\mathscr{P}_{s,n}. \end{aligned} \tag{4.22}$$

With $\beta_{i,n}$ the sum of setup periods $\sigma_{g_i,n}$ and green periods $\tau_{i,p}$, $p\neq n$ (for example, $\beta_1=\alpha_5+\sigma_{g_5,1}$), it can be seen that equation (4.22) is quadratic in the optimization variables $\tau_{i,n}^\mu$ and $\tau_{i,n}^\lambda$, provided that cycle time T is a constant. Hence, to derive the optimum for a given sequence using QP, the solution of the QP must be derived for all cycle times in the range (4.6). However, by deriving the minimal required cycle time T^* for this sequence, using a LP as shown above, the range can be reduced if $T^{\min} < T^*$ or the sequence can be discarded if $T^{\max} < T^*$. Next, the optimum is found by minimizing the objective function over the range $\min(T^{\min}, T^*) \leq T \leq T^{\max}$. Usually, this optimum is located at the lower bound of T and can be determined by solving the QP twice, i.e., if $J(\max(T^{\min}, T^*)) < J(\max(T^{\min}, T^*) + \varepsilon)$ where $J(T)$ is the minimal objective value as function of cycle time T. Otherwise, an algorithm, e.g., the bisection method, can be used to locate the optimum.

4.5.3 Constraints

We have already introduced constraints on green periods (4.5), cycle time (4.6) and stability (4.15), which have to be taken into account. The maximal queue content restriction, see (4.1), can be formulated as

$$x_{i,n} \leq x_n^{\max} - \lambda_n \sigma_{g_{i-1},n}, \qquad \forall n \in g_i, \quad i = 1, 2, ..., |s|,$$

since the maximal content of a queue is reached at the moment right before serving that queue.

The green period consists of a part serving at maximal rate and a part serving at arrival rate:

$$\tau_{i,n} = \tau_{i,n}^{\mu} + \tau_{i,n}^{\lambda}, \qquad \forall i \in \mathscr{P}_{s,n}, \quad n = 1, 2, ..., N.$$

The time serving flow n at rate μ_n is bounded by the time needed to empty queue $x_{m,n}$, as backlog is not allowed. Therefore,

$$\tau_{i,n}^{\mu} \leq \frac{x_{i,n}}{\mu_n - \lambda_n}, \qquad \forall i \in \mathscr{P}_{s,n}, \quad n = 1, 2, ..., N.$$

Moreover, all queue contents and green periods are nonnegative. Furthermore, as periodic behavior is considered for stable systems, the queue lengths for all queus at the start and the end of a cycle are identical, i.e., $x_n(0) = x_n(T)$, $n = 1, 2, ..., N$. Due to the topology of the intersection, it is conceivable that the setup period for switching from group g_1 to group g_2 exceeds the total time required for switching from group g_1 to group g_2 via other groups. Consequently, it is possible that for switching from group g_1 to group g_2 via other groups, the cars in group g_2 start driving while the cars in conflicting group g_1 have not cleared the intersection yet, which is not allowed. Therefore, switching between groups g_1 and g_2 via other groups should take at least as much time as the setup period from group g_1 to group g_2, i.e., σ_{g_1,g_2}. For ease of exposition, we present the constraint that the time required for switching from group g_1 to g_2 via g_3 has a lower bound, given by the setup period required for switching between groups g_1 and g_2:

$$\sigma_{g_1,g_2} \leq \alpha_{g_3} + \sigma_{g_3,g_2}.$$

Similarly, we restrict the duration of all possible trajectories between two groups by imposing the constraint

$$\sigma_{g_i,g_j} \leq \sigma_{g_{j-1},g_j} + \sum_{k=i+1}^{j-1} \alpha_k, \qquad \forall i, j = 1, 2, ..., N, \quad i \neq j. \tag{4.23}$$

Note that constraint (4.23) specifically holds for signalized traffic intersections. For manufacturing systems, for instance, the setup periods are not linked to fluid leaving the system and therefore constraint (4.23) is redundant.

4.5.4 Generic optimization problem

The generic optimization problem, given sequence s, for the different objectives reads as follows

$$\min \quad J, \qquad\qquad J \in \{J_w, J_\varphi, \chi, T\}$$

$$\begin{aligned}
s.t. \qquad & \alpha_i = \sigma_{g_{i-1},n} + \tau^{\mu}_{i,n} + \tau^{\lambda}_{i,n}, && \forall n \in g_i, \quad i = 1,2,\dots,|s|, \\
& T = \sum_{i=1}^{|s|} \alpha_i, && \\
& \tau_{i,n} = \tau^{\mu}_{i,n} + \tau^{\lambda}_{i,n}, && \forall i \in \mathscr{P}_{s,n}, \quad n = 1,2,\dots,N, \\
& x_{i,n} = \lambda_n \left(T - \tau^{\mu}_{i,n} - \sigma_{g_{i-1},n} - \sum_{j \in \mathscr{P}_{s,n}} \tau^{\lambda}_{j,n} \right) - && \\
& \qquad - \mu_n \left(\sum_{j \in \mathscr{P}_{s,n} \setminus i} \tau^{\mu}_{j,n} \right), && \forall i \in \mathscr{P}_{s,n}, \\
& \tau^{\mu}_{i,n} \leq \frac{x_{i,n}}{\mu_n - \lambda_n}, && \forall i \in \mathscr{P}_{s,n}, \quad n = 1,2,\dots,N, \\
& \tau^{\min}_n \leq \sum_{i \in p} \tau_{i,n} \leq \tau^{\max}_n && \forall p \in P_{s,n}, \quad n = 1,2,\dots,N, \\
& x_{i,n} \leq x^{\max}_n - \lambda_n \sigma_{g_{i-1},n}, && \forall n \in g_i, \quad i = 1,2,\dots,|s|, \\
& x_{i,n} \leq \chi - \lambda_n \sigma_{g_{i-1},n} && \text{if } J = \chi, \\
& \rho_n T \leq \sum_{i \in g} \tau_{i,n}, && \forall g \in \mathscr{P}_{s,n}, \quad n = 1,2,\dots,N, \\
& \sigma_{g_i,g_j} \leq \sigma_{g_{j-1},g_j} + \sum_{k=i+1}^{j-1} \alpha_k, && \forall i,j = 1,2,\dots,N, \quad i \neq j,
\end{aligned}$$

in which the optimization variables are given by

$$\tau^{\mu}_{i,n},\ \tau^{\lambda}_{i,n} \qquad \forall n \in g_i, \quad i = 1,2,\dots,|s|.$$

4.6 Illustrations

The intersection presented in Figure 4.2 is optimized below with respect to the weighted average amount of users J_w (vehicles, pedestrians and bicyclists). For this intersection, we consider the parameters

$$\Sigma = \begin{bmatrix} 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 2 & 6 & 0 & 5 & 0 \\ 3 & 0 & 3 & 0 & 5 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \lambda = \begin{bmatrix} 3 \\ 0.1 \\ 1 \\ 0.1 \\ 4 \end{bmatrix}, \quad \mu = \begin{bmatrix} 10 \\ 2 \\ 10 \\ 2 \\ 10 \end{bmatrix},$$

where Σ is the matrix with setup periods, i.e., switching from flow n to flow m requires setup period $\Sigma_{nm} = \sigma_{n,m}$. The weight factors for all queues are 1. Restrictions on the cycle time, minimal green periods and maximal queue contents are given by:

$$40 \leq T \leq 60, \tag{4.24a}$$
$$\tau^{\min} = \begin{bmatrix} 3 & 6 & 3 & 6 & 3 \end{bmatrix}, \tag{4.24b}$$
$$x_n^{\max} = 100, \qquad n = 1,2,...,N. \tag{4.24c}$$

Using these parameters, we present a QP formulation for a sequence with three groups. Next, the optimal periodic behavior of this intersection is investigated.
As an example, consider the sequence $s = (\{1,2,5\},\{4\},\{3\})$. The QP formulation for this sequence is presented below. Each flow is contained in a single group and therefore has a single green period. For ease of exposition, the group labels are omitted, i.e., $\tau_{m,n} = \tau_n$. Note that constraints (4.23) are omitted, due to the setup periods and minimal green periods these constraints can not be violated.

$$\begin{aligned} \min_{\tau_1,\tau_2,\tau_3,\tau_4,\tau_5} \quad & \frac{c_i\lambda_i}{2}(\alpha_2+\alpha_3+\sigma_{3,i})(T-\tau_i^{\lambda}) + \frac{c_3\lambda_3}{2}(\alpha_1+\alpha_2+\sigma_{4,3})(T-\tau_3^{\lambda}) + \\ & \frac{c_4\lambda_4}{2}(\alpha_1+\alpha_3+\sigma_{\{1,2,5\},4})(T-\tau_4^{\lambda}), \qquad i = 1,2,5 \end{aligned}$$

$$\begin{aligned} s.t. \qquad \alpha_1 &= \sigma_{3,n} + \tau_n^{\mu} + \tau_n^{\lambda}, && n = 1,2,5, \\ \alpha_2 &= \sigma_{\{1,2,5\},4} + \tau_4^{\mu} + \tau_4^{\lambda}, \\ \alpha_3 &= \sigma_{4,3} + \tau_3^{\mu} + \tau_3^{\lambda}, \\ T &= \alpha_1 + \alpha_2 + \alpha_3, \\ \tau_n &= \tau_n^{\mu} + \tau_n^{\lambda}, && n = 1,2,3,4,5, \\ \tau_n^{min} &\leq \tau_n, && n = 1,2,3,4,5, \\ \rho_n T &\leq \tau_n, && n = 1,2,3,4,5, \\ \tau_n^{\mu} &\leq \frac{\lambda_n}{\mu_n - \lambda_n}(\alpha_2+\alpha_3+\sigma_{3,n}), && n = 1,2,5, \\ \tau_3^{\mu} &\leq \frac{\lambda_3}{\mu_3 - \lambda_3}(\alpha_1+\alpha_2+\sigma_{4,3}), \\ \tau_4^{\mu} &\leq \frac{\lambda_4}{\mu_4 - \lambda_4}(\alpha_1+\alpha_3+\sigma_{\{1,2,5\},4}). \end{aligned}$$

For this intersection a simple schedule is desired, therefore a sequence is limited to contain maximally four groups, i.e., $G = 4$. The objective is to minimize the average amount of users (vehicles, pedestrians and cyclists) on the intersection. For simplicity, all weights are equal to 1. First, we consider finding an optimal solution for $\Theta = 1$, i.e., all flows receive a single green period in a cycle. The total number of feasible sequences satisfying the constraints is 282. The five sequences with lowest average queue contents are presented in Table 4.1. For each sequence, denoted by s_j, $j = 1,2,...,5$, the composition of groups are given, labeled by g_i with $i = 1,2,...,|s|$.

	g_1	g_2	g_3	g_4
s_1	1, 2, 5	2, 5	4	3, 5
s_2	1, 2, 5	5	4	3, 5
s_3	1, 2, 5	2, 4	4	3, 5
s_4	1, 2, 5	2	4	3, 5
s_5	1, 2, 5	4	3	3, 5

Table 4.1: Optimal sequences for $\Theta = 1$ and $G = 4$.

The minimal average queue content W, corresponding cycle time T and service periods of groups α_j for these sequences are presented in Table 4.2.

	J_w	T	α_1	α_2	α_3	α_4
s_1	49.36	41.77	25.59	2.00	7.00	7.18
s_2	49.47	41.85	25.67	2.00	7.00	7.19
s_3	53.24	45.49	28.94	4.00	5.00	7.55
s_4	53.29	45.52	28.97	3.00	6.00	7.55
s_5	53.43	45.63	29.07	9.00	5.00	2.56

Table 4.2: Optimal results for each sequence, $\Theta = 1$ and $G = 4$.

This example illustrates that using maximal groups only is not optimal in this case, i.e., it can be seen that the optimal periodic behavior, described by sequence s_1, does not consist of maximal groups only. In this case, group g_2 is a subset of g_1. This occurs, since flow 1 has the largest setup period for setting up from group g_1 to group g_3, i.e. $\sigma_{1,4} > \sigma_{2,4}$ and $\sigma_{1,4} > \sigma_{5,4}$. Therefore, while the cars from flow 1 are clearing the intersection to make room for flow 4, flows 2 and 5 can both have additional green periods (2 seconds in group g_2). Since $\sigma_{5,4} > \sigma_{2,4}$, the performance would increase if an additional group is added between g_2 and g_3 consisting of flow 2. However, this is not possible due to the upper bound on the number of groups. Note that adding these groups does not affect the cycle time, as the initial red period of the subsequent group would decrease.

Next, let us investigate optimal behavior for sequences with $\Theta = 2$. Note that this is also the upper bound, which follows from (4.3). The total number of feasible sequences is 486. The five sequences with minimal average queue content are presented in Table 4.3. It can be seen that in all these sequences flows 1 and 5 are served in groups g_1 and g_3 with groups containing either flow 3 or flow 4 in between. Intuitively, it makes sense to serve the flow with the highest load as much as possible. Also, flow 5 is sometimes served in three groups, as it conflicts only with flow 4. Note that for the best sequence s_1, not every group is a maximal group, i.e., g_3 is a subset of $\{1,2,5\}$.

	g_1	g_2	g_3	g_4
s_1	1, 2, 5	2, 4	1, 5	3, 5
s_2	1, 2, 5	4	1, 5	3, 5
s_3	1, 5	2, 4	1, 5	3, 5
s_4	1, 5	3, 5	1, 5	2, 4
s_5	1, 2, 5	2, 4	1, 5	3

Table 4.3: Optimal sequences for Θ = 2 and $G = 4$.

Table 4.4 presents the optimization results for the sequences presented in Table 4.3. It can be seen that the optimal solution, with $\Theta = 2$, has decreased the average queue content by 8.8% compared to the best solution for $\Theta = 1$.

	J_w	T	α_1	α_2	α_3	α_4
s_1	45.35	44.31	20.05	9.00	8.14	7.12
s_2	45.90	44.67	20.51	9.00	8.14	7.02
s_3	46.23	42.13	17.90	9.00	8.14	7.09
s_4	46.24	41.00	14.33	6.85	10.81	9.00
s_5	47.54	42.21	14.66	9.00	14.33	4.22

Table 4.4: Optimal results for each sequence, Θ = 2 and $G = 4$.

Omitting constraint (4.24a) gives that the minimal cycle time is $T^* = 26$ and corresponding performance is $W = 64.29$. Note that the cycle time corresponding to the optimal schedule is larger than the minimal required cycle time. Therefore, at least one flow receives a green signal while the queue content is zero, i.e., at least one group has a slow-mode.

The queue levels of all flows for sequence s_1 are presented in Figure 4.7. The upper figure shows the queue levels of the vehicle flows and the pedestrian flows are presented in the figure in the middle. The service schedule is presented in the lower figure. It can be seen that in this sequence, all flows receive a green signal while the queue is empty, i.e., all queues have a slow-mode. Therefore, all queues can handle additional arrivals. If the arrivals increase by $\begin{bmatrix}74 & 1040 & 60 & 270 & 71\end{bmatrix}$%, for flows 1-5 respectively, the intersection operates at full capacity. This also indicates that this sequence can handle some fluctuations in the arrivals. Additionally, note that only the duration of group 2 is determined by the minimal green period constraint of flow 4. Also, the restrictions on cycle time are inactive, where the minimal cycle time (with green period constraints) is 26 seconds.

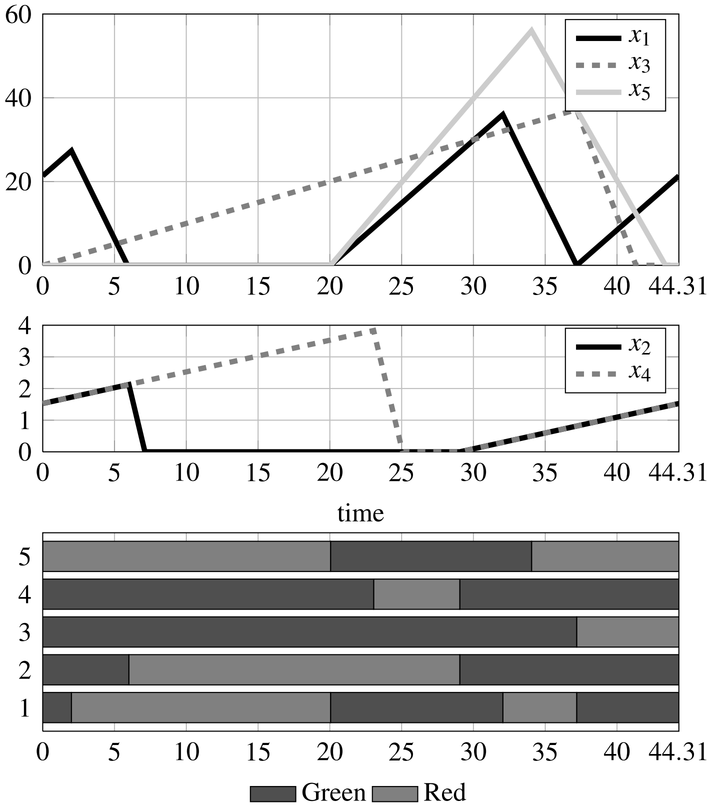


Figure 4.7: Queue levels of vehicle flows 1,3 and 5 (upper), pedestrian flows 2 and 4 (middle) and corresponding service schedule (lower) for sequence s_1 in Table 4.4.

Remark 4.6.1 (Fluctuating arrival rates)**.** *In this chapter we assume constant arrival rates, whereas in practice the arrival rates may fluctuate. Nevertheless, slow-modes in the schedule create extra capacity that can, for instance, cope with arrival fluctuations, as the results in Figure 4.7 also indicate. The presence of slow-modes can be caused by requiring minimal green periods, which exceed the minimal required green periods, requiring the minimal cycle time to be longer than necessary to serve all incoming vehicles. It can also be the result of a trade-off between losing capacity due to a slow-mode or losing capacity due to spending a relative larger amount of time on setups. Moreover, a robust schedule can also be derived by considering larger arrival rates.*

By varying the number of maximal groups G and Θ we get the following optimal periodic results, presented in Table 4.5. The number between brackets indicates the total number of feasible sequences corresponding to the problem. Note that the results correspond to the sequence with exactly G groups and at least one flow has Θ green periods in a cycle. Sequence $(\{1,2,5\},\{2,4\},\{1,5\},\{3,5\},\{1,5\},\{3,5\})$

consisting of 6 groups and $\Theta = 3$ has a 38.57% better performance, i.e., $J_w = 35.62$ compared to the optimal sequence with $G = 4$ and $\Theta = 1$. It can be seen that scheduling multiple green periods for some flows improves performance.

$\Theta \backslash G$	3	4	5	6
1	53.43 (18)	49.36 (282)	49.27 (1416)	49.27 (3382)
2	-	45.35 (204)	40.33 (6168)	37.26 (78950)
3	-	-	-	35.62 (12744)

Table 4.5: Optimal results J_w^*, varying Θ and G. The number of feasible sequences are indicated between brackets.

For the system with average waiting time as performance criteria, the optimal sequence is $\{\{1,2,5\},\{2,4\},\{1,5\},\{3,5\},\}$ and minimal average waiting time $J_\varphi^* = 44.31$. If the queue lengths are minimized, the optimal sequence is given by $\{\{1,2,5\},\{5\},\{4\},\{3,5\},\}$ and minimal queue length $\chi^* = 54$.

4.6.1 Three-queue manufacturing system

To illustrate the presented method for deriving optimal periodic behavior for other types of systems, the optimal periodic behavior of a three-queue manufacturing system is discussed below. A single server with three queues is considered, see Figure 4.8. The arrival and service rates are indicated in the figure, all setup periods have a duration of 1 and all costs are equal (we consider $c_1 = c_2 = c_3 = c_4 = 1$). Furthermore, the server is restricted to serve only one queue at a time.

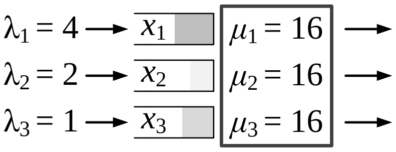


Figure 4.8: Three-queue switching server.

In [63] a theory is developed to compute a lower bound on the mean work in progress, i.e., average queue contents, for a server with $i \in \mathbb{N}$ queues. This lower bound is often not reachable in practice, i.e., for symmetrical systems (identical arrival rates, service rates and setup periods for all queues) the theoretical lower bound can be achieved, but for all other situations this might not be the case. However, this lower bound is a good reference point. For the system depicted in Figure 4.8, the theoretical lower bound according to [63] is 14.3875 and the number of setups per time unit for queues 1, 2 and 3 are 0.242, 0.185 and 0.135 respectively. This ratio is roughly 4:3:2. In [103], the performance of sequences with this ratio and several other ratios is determined. Using the method presented in this chapter, the optimal periodic behavior of this system is derived given the constraints $G = 9$ and $\Theta = 4$.

The four best sequences are presented in Table 4.6, minimizing the average queue length. The optimal sequence, s_1, has a 3:3:2 ratio and is a fraction higher than the theoretical lower bound. Note that, for this example, allowing multiple service periods to queues within a cycle results in better performance, i.e., compare J_w for sequence s_4, where each queue is served once, to the J_w of the other sequences.

	Sequence	J_w
s_1	(1, 2, 3, 1, 2, 1, 3, 2)	14.897
s_2	(1, 2, 1, 3)	14.926
s_3	(1, 3, 2, 1, 2, 1, 3, 1, 2)	15.034
s_4	(1, 2, 3)	15.167

Table 4.6: Optimal sequences for $\Theta = 4$ and $G = 9$.

4.7 Case study

To illustrate that this method is computationally affordable for real-life intersections, we illustrate the derivation of the optimal periodic behavior of a large real-life intersection. We consider the traffic intersection presented in Figure 4.9, which is the most complicated intersection of the city Eindhoven, the Netherlands. This intersection is a combination of a flared and channelized intersection between the John F. Kennedylaan and Onze Lieve Vrouwestraat and is part of the main arterial around the center of Eindhoven. Daily, almost 30000 motor vehicles cross this intersection, see [20, 22]. In reality it contains 10 car signals and 16 signals for cyclists and pedestrians. By taking certain nonconflicting signals together, this can be reduced to seven car signals (flows 1–7) and four pedestrian/cyclist signals (flows 8–11).

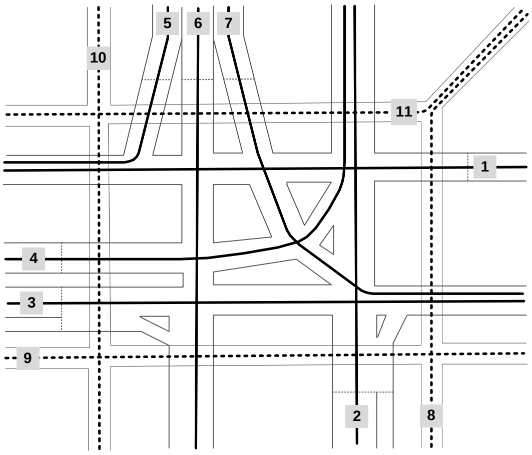


Figure 4.9: Traffic intersection the city in Eindhoven, the Netherlands.

Arrival and saturation rates in vehicles per second during an evening rush-hour, together with the weights of each flow, are presented in Table 4.7.

n	1	2	3	4	5	6	7
c_n	0.1210	0.1580	0.1530	0.1750	0.0680	0.2030	0.1190
λ_n	0.0731	0.0956	0.0922	0.1058	0.0411	0.1228	0.0717
μ_n	0.4722	0.4722	0.4722	0.4722	0.4722	0.4722	0.4722

Table 4.7: Weights, arrival and saturation rates of the vehicle flows, see [22].

The number of pedestrians and cyclists crossing the intersection is far less than the number of vehicles, $\lambda_n = 0.01$, $\mu_n = 0.4722$ for $n = 8, 9, 10, 11$. Also the weight factor of the pedestrians and cyclists is lower ($c_n = 1e^{-4}$). The bicycle and pedestrians flows do not conflict. Instead of drawing a complex graph, we introduce the confliction matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $A_{i,j} = 1$ indicates that flows i and j conflict and $A_{i,j} = 0$ otherwise. Furthermore, setup periods in seconds are given by

$$\Sigma = \begin{bmatrix} 0 & 0 & 0 & 0 & 8 & 6 & 5 & 0 & 0 & 10 & 0 \\ 5 & 0 & 7 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 10 \\ 0 & 7 & 0 & 0 & 0 & 1 & 3 & 10 & 0 & 1 & 0 \\ 9 & 6 & 0 & 0 & 0 & 3 & 4 & 0 & 0 & 1 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 6 & 4 & 0 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 5 & 3 & 2 & 0 & 0 & 0 & 9 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 7 & 7 & 7 & 0 & 0 & 0 & 0 \end{bmatrix},$$

When addressing stability of the intersection, the flows are divided into groups which can be served simultaneously, i.e. non-conflicting flows. The total number of groups for this system is 72. The maximal allowed groups are given by

$$\begin{aligned}\mathscr{G}^{\max} = &\{\{5,6,7\},\{2,5,6,8\},\{3,4,5,9\},\{5,7,9\},\{1,3,9,11\},\{2,6,8,10\},\{6,7,10\},\\ &\{7,9,10\},\{4,5,8,9\},\{8,9,10,11\}\}.\end{aligned}$$

The maximal cliques are given by

$$\begin{aligned}\mathscr{C}^{\max} = &\{\{2,4,7,11\},\{1,2,4,7\},\{4,6,11\},\{3,7,8\},\{2,3,7\},\{1,7,8\},\{1,5,10\},\\ &\{1,4,10\},\{1,4,6\},\{6,9\},\{5,11\},\{3,10\},\{3,6\},\{2,9\}\}\end{aligned}$$

Using these groups, the stability conditions for the unconstrained system, derived by (4.11), are the following

$$\rho_{11}+\rho_2+\rho_4+\rho_7 \leq 1, \tag{4.25a}$$
$$\rho_1+\rho_2+\rho_4+\rho_7 \leq 1, \tag{4.25b}$$
$$\rho_3+\rho_7+\rho_8 \leq 1, \tag{4.25c}$$
$$\rho_1+\rho_{10}+\rho_5 \leq 1, \tag{4.25d}$$
$$\rho_6+\rho_9 \leq 1, \tag{4.25e}$$
$$\rho_{11}+\rho_4+\rho_6 \leq 1, \tag{4.25f}$$
$$\rho_1+\rho_7+\rho_8 \leq 1, \tag{4.25g}$$
$$\rho_2+\rho_9 \leq 1, \tag{4.25h}$$
$$\rho_2+\rho_3+\rho_7 \leq 1, \tag{4.25i}$$
$$\rho_{11}+\rho_5 \leq 1, \tag{4.25j}$$
$$\rho_1+\rho_4+\rho_6 \leq 1, \tag{4.25k}$$
$$\rho_1+\rho_{10}+\rho_4 \leq 1, \tag{4.25l}$$
$$\rho_3+\rho_6 \leq 1, \tag{4.25m}$$
$$\rho_{10}+\rho_3 \leq 1, \tag{4.25n}$$

$$\rho_{10}+\rho_{11}+\rho_2+\rho_3+\rho_5+\rho_7 \leq 2, \tag{4.25o}$$
$$\rho_{10}+\rho_{11}+\rho_3+\rho_5+\rho_6 \leq 2 \tag{4.25p}$$

where the first part, equations (4.25a)-(4.25n) are conditions based on the cliques and the conditions (4.25o)-(4.25p) are based on whole intersection graph. With the parameters from Table 4.7, the stability conditions (4.25) are met. Therefore, there exists a schedule that results in a stable system if constraints on cycle time and service periods are omitted.

The constraints on cycle time and green periods are given by

$$\begin{aligned}&90 \leq T \leq 110, && \\ &\quad 5 \leq \tau_n, && \text{if } n = 1,2,\ldots,7,\\ &\quad 13 \leq \tau_n, && \text{if } n = 8,9,10,11.\end{aligned}$$

Note that the minimal green periods for the pedestrian/cycle lanes are 13 seconds and these lanes intersect with the vehicle lanes. Therefore, although the arrival rates and weights are low, these flows can not be ignored. In [22], a schedule of green periods used for the current situation is given, which aims at minimizing the average waiting time. Using (4.19), this schedule results in an average waiting time of $\Phi = 29.45$ seconds.

The five sequences with best performance for a schedule with $G = 4$ and $\Theta = 1$ are presented in Table 4.8 together with their optimal results presented Table 4.9. Using the optimal schedule, average waiting time can be decreased by 17.0%. Note that the results for both sequences s_1 and s_2 and sequences s_4 and s_5 are similar. However, the presence of flow 9 in group g_1 results in better performance, but it is hardly visible due to the low costs.

	g_1	g_2	g_3	g_4
s_1	1, 3, 9, 11	3, 4, 5, 9	2, 5, 6, 8	6, 7, 10
s_2	1, 3, 11	3, 4, 5, 9	2, 5, 6, 8	6, 7, 10
s_3	1, 3, 9, 11	3, 4, 5	2, 5, 6, 8	6, 7, 10
s_4	1, 3, 9, 11	3, 4, 9	2, 5, 6, 8	6, 7, 10
s_5	1, 3, 11	3, 4, 9	2, 5, 6, 8	6, 7, 10

Table 4.8: Best sequences for the intersection in Figure 4.9, $G = 4$ and $\Theta = 1$.

	J_Φ	T	α_1	α_2	α_3	α_4
s_1	25.18	90.00	14.00	27.06	33.94	15.00
s_2	25.18	90.00	14.00	27.06	33.94	15.00
s_3	25.57	90.00	19.00	23.33	32.67	15.00
s_4	25.82	90.00	14.00	22.73	38.27	15.00
s_5	25.82	90.00	14.00	22.73	38.27	15.00

Table 4.9: Optimal results for sequences in Table 4.8.

Allowing multiple green periods, i.e., $\Theta = 2$, results in better performance, see the results presented in Tables 4.10 and 4.11. For all sequences s_1–s_5 it can be seen that both flow 6, which is the flow with the highest load and cost, and flow 3 receive two green periods in a cycle. The optimal periodic behavior lowers the performance by 20.9% compared to the schedule in [22]. Note that the sequence s_1 for the optimal periodic behavior does not consist of maximal groups only. Group g_1 is a subset of $\{1,3,9,11\}$. Adding flow 9 into group g_1 decreases the performance by elongating the time spent in this group, i.e., $\sigma_{6,9} = 6$ and $\tau_9^{\min} = 13$ results in $\alpha_1^{\min} = 19$.
Also note that for all presented sequences, the lower bound constraint on the cycle time is active. Removing this constraint would result in an weighted average wait-

ing time of $\Phi = 18.26$ for sequence $\{1,3,9,11\},\{3,4,5,9\},\{2,6,8,10\},\{5,6,7\}$ and $T = 60.89$.

	g_1	g_2	g_3	g_4
s_1	1, 3, 11	2, 5, 6, 8	3, 4, 5, 9	6, 7, 10
s_2	1, 3, 11	2, 6, 8, 10	3, 4, 5, 9	5, 6, 7
s_3	1, 3, 9, 11	2, 5, 6, 8	3, 4, 5, 9	6, 7, 10
s_4	1, 3, 9, 11	2, 5, 6, 8	3, 4, 5	6, 7, 10
s_5	1, 3, 9, 11	2, 6, 8, 10	3, 4, 5, 9	5, 6, 7

Table 4.10: Best sequences for the intersection in Figure 4.9, $G = 4$ and $\Theta = 2$.

	J_Φ	T	α_1	α_2	α_3	α_4
s_1	24.37	90.00	14.00	30.10	28.24	17.66
s_2	24.47	90.00	14.00	25.21	30.25	20.53
s_3	24.70	90.00	19.00	27.29	26.05	17.66
s_4	24.70	90.00	19.00	27.30	26.04	17.66
s_5	24.74	90.00	19.00	25.21	28.13	17.66

Table 4.11: Optimal results for sequences in Table 4.10.

Remark 4.7.1 (Computation times)**.** *For our case study we used MATLAB on a 3.2GHz Intel i5 CPU. Considering the case where $\Theta = 2$ and $G = 4$, computation time for generating all feasible groups (72) was 0.1 seconds, computation time for generating all feasible sequences (2730) was 34.6 seconds and computation time for deriving the optimal solution was 63.1 seconds.*

4.8 Summary

This chapter presents a method to derive an optimal periodic signal timing plan for a multi-queue switching server. In addition to the periodic behavior of two-queue servers, presented in Chapter 3, multiple queues can be served simultaneously and queues can be served multiple times in a cycle. These systems can represent a class of queueing networks. Isolated signalized traffic intersections with fixed schedules is one of these queueing networks, and periodic behavior of these networks has been investigated. Instead of determining the number of groups, composition of groups and order of groups in a sequence *a priori*, we solve a LP or QP optimization problem for every feasible sequence, i.e., computing green periods and cycle time. This includes sequences, which serve some flows multiple times and serve some flows more than others. The problem is attacked in two separate steps.

In the first step, all feasible sequences are derived. This requires the generation of all groups, i.e, finding all possible combinations of flows which are compatible.

Note that for all possible groups, also groups which are subsets of the maximal allowed groups are considered, which sometimes can lead to an optimal periodic solution, as shown by an example. Second, all possible combinations of groups are constructed, regarding the constraint on the maximal number of groups. Third, all feasible sequences are derived from these combinations. Scheduling some flows more than others, i.e., giving multiple green periods to some flows, which has not been addressed before, resulted in significant better performance for all examples discussed.

The second step consists of deriving the optimal schedule, i.e., green period durations, for each possible sequence. These schedules are calculated separately with respect to restrictions on cycle time, green periods and maximal queue content, similar as in Chapter 3 for two-queue servers. From these solutions, the best one is selected, which is the optimal periodic behavior for that intersection, given the restrictions on total number of groups in a sequence and maximal number of green periods for a flow. We show that the problem of finding the optimal green periods, given a sequence, can be formulated as a QP or LP. This creates flexibility in setting the constraints and objectives.

Due to all the calculations in each step, one might question the computational feasibility in order to reach the optimal periodic solution. This certainly has its limitations, but as a case study, we considered a complex intersection in Eindhoven consisting of 11 different flows and the resulting computational effort was satisfactory.

For a network of switching servers, fluid flows from one server to another server and therefore the arrival rates at the queues are non-constant. Therefore, periodic behavior of networks of switching servers is investigated in Chapter 5. Furthermore, multi-queue switching servers are also considered in Chapter 7. Here, an observer is derived for a multi-queue switching server with a specific policy. Based on the measured arrival rates and partial information of the output rates, the queue contents are reconstructed.

Chapter 5

Periodic behavior of a network of switching servers

Periodical behavior of single switching servers has been studied for two-queue servers in Chapter 3 and for multi-queue servers in Chapter 4. In this chapter networks of switching servers are considered. Here, fluid flows from one server to another server and therefore the arrival rates at some queues are non-constant, which differs from queues at single server systems. Examples of networks of switching servers are manufacturing systems where flows of goods move through the factory and require operations by multiple machines, or networks of signalized intersections, where flows of vehicles move from one intersection to another.

Fluid flows through the network of multi-queue switching servers via multiple predefined routes. Each fluid flow originates from an external source and leaves the network after visiting (multiple) servers. At each server visit, the fluid is stored in a dedicated queue. As presented in Chapter 4, multi-queue switching servers can serve multiple queues and switching between service of queues takes time. In the network, fluid travels between queues, which leads to *piecewise constant* arrival rates at queues, i.e., the piecewise constant departure rate from a queue is the arrival rate at the downstream queue. This feature considerably extends the work presented in Chapter 4. Again, we consider optimal periodic behavior, or the optimal periodic scheduling problem, of these queueing networks.

By dividing for each queue its service period into multiple *phases*, characterized by the different service rates, we are able to derive optimal schedules for a network of switching servers with instant transportation of fluid, i.e., fluid from the queue attended by a server immediately enters the successive queue. Here it is assumed that all servers have identical cycle times and that switching moments are *synchronized*. Queueing networks with non-negligible transportation delays, such as e.g., networks of signalized traffic intersections, where traveling between intersections takes time, are not considered. Similar to the optimization problem of Chapters 3

This chapter is partly based on [105].

and 4, the optimization of service and idle periods of all queues in a given sequence is formulated as a linear or quadratic programming problem. This formulation allows considerable flexibility in the constraints and performance criteria.

The optimal scheduling problem is decomposed into two consecutive steps, similar to the approach presented in Chapter 4. The first step consists of generating all feasible sequences, i.e., forming groups of queues that can be served simultaneously and combining these groups into sequences, satisfying the constraints. In the second step, the optimal periodic service schedule, for which the queue lengths at the start and end of a cycle are equal, is derived for each sequence.

In Section 2.2, an introduction of a network of switching servers has been already presented. The remainder of this chapter is organized as follows. Section 5.1 introduces the queueing network model, a well-known two-server network that will serve as a running example, the service schedule and it introduces the problem. Section 5.2 presents a brief description on the sequence generation, the preparation stage. The optimization stage for a given sequence, based on the performance criteria, is presented in Section 5.3. Section 5.4 presents examples of optimal schedules. A summary is provided in Section 5.5.

5.1 System description

In this section we introduce the model, specify the service schedule along with constraints and conclude with the problem formulation.

In a fluid flow queuing network, fluid flows from one server to another, and eventually leaves the network. The network consists of multi-queue switching servers, labeled by $j = 1,2,...,S$. Fluid flows through the network via a single route or multiple predefined routes. A route, starting from an external source, indicates which servers are visited and also in which order. Each server visited along a route has a specific queue dedicated to the fluid following this route. Therefore, each queue has a *single* source, i.e., fluid arrives at a queue either from an external source, or after being served at the preceding queue on the same route. The queues are labeled in the order of occurrence along the route(s), by $n = 1,2,...,N$. Queues share the capacity of a server, but in this chapter we assume that the server can serve only *one queue at a time*. This is common behavior for, e.g., manufacturing networks, where a robot can perform different tasks on several queues. However, for networks of signalized traffic intersections, where at each intersection multiple flows can receive green lights simultaneously (depending on the layout of the intersection), the servers can serve multiple queues simultaneously. This feature can be added, as presented in Chapter 4, but it is omitted for ease of reading. The set of queues exclusively served by server j is denoted by $Z(j)$. So $Z(1),...,Z(S)$ is a partitioning of the set of all queues. The arrival rate at queue n is denoted by $\lambda_n(t)$, which is constant if fluid arrives from an external source or piecewise constant if fluid arrives from an upstream queue along the route. Once attended by the server, queue n is maximally served at a queue-dependent deterministic rate μ_n. Note that the departure rate from queue n, which is the arrival rate to queue $n+1$, can be less than μ_n, e.g., if queue n is attended by the server and $x_n = 0$, then $\lambda_{n+1}(t) = \max(\lambda_n(t), \mu_n)$. Switching of the server from queue $k \in Z(j)$ to queue $l \in Z(j)$, implies a *setup period* with setup period $\sigma_{k,l}$, which can not be interrupted. Note that $\sigma_{k,k} = 0$.

Also, fluid transportation between queues is assumed to be instant, i.e., while being served, fluid instantaneously leaves the network or instantaneously arrives at the downstream queue. For networks of manufacturing systems, the transportation time in general is small compared to the service and setup periods and can therefore be neglected. Due to this assumption, the problem can be formulated as a LP or QP problem. However, this restricts the class of considered networks, as some networks have non-negligible transportation times, such as networks of traffic intersections, i.e., vehicles require a certain time driving from one intersection to the other. For these networks, a heuristic to obtain a good service schedule, based on optimal periodic behavior of single multi-queue servers (Chapter 4) is presented in Section 9.1.

As illustrative and running example, we present a well-known manufacturing network, as introduced in [62] and also discussed in Section 2.2. Consider the two server queueing network presented in Figure 5.1. Server 1 serves queues 1 and 4 and server 2 serves queues 2 and 3, i.e., $Z(1) = \{1,4\}$ and $Z(2) = \{2,3\}$. The network processes a single fluid flow arriving from an external source with constant rate $\lambda_1(t) = \lambda$ at queue 1. The fluid flow consecutively visits servers 1, 2, 2, and 1 via queues 1, 2, 3, and 4 respectively. Queue 1 is the only queue receiving fluid from an external source, the other queues receive the fluid from preceding queues. Switching between service of queues 1 and 4, and between service of queues 2 and 3 requires a setup period.

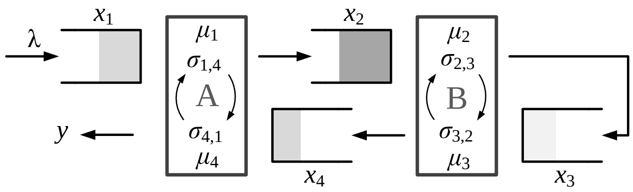


Figure 5.1: Two server, four queue network, introduced in [62].

5.1.1 Service schedule

The service schedule for a network of switching servers is similar to the service schedules of single multi-queue switching servers, presented in Section 4.2. A short description is presented here and differences between de schedules are indicated.

The queues that can be served simultaneously are divided into groups $g \in \mathcal{G}$. Not assigning a queue from a server to a group results in undesired idling of that server and is therefore not allowed. Also, each server can serve a single queue. Therefore, each server assigns one of its queues, i.e., $|g| = S$. The maximal number of groups in a sequence s has an upper bound G, i.e.,

$$|s| \leq G. \tag{5.1}$$

The number of service periods θ_s received by a queue in sequence s can not exceed the maximal number of service periods Θ received by a specific queue during a cycle, i.e.,

$$\theta_s \leq \Theta. \tag{5.2}$$

A sequence is *feasible* if all queues are served at least once and if constraints (5.1)–(5.2) are satisfied. The set of feasible sequences is denoted by $\mathscr{S}$. For example, a feasible sequence for the network depicted in Figure 5.1 is $(\{1,2\},\{3,4\})$, i.e., the sequence consists of two groups, $g_1 = \{1,2\}$ and $g_2 = \{3,4\}$, which are successively served. Serving group $\{1,2\}$ indicates serving queues 1 and 2 simultaneously, each by its own server.

The time it takes to serve all groups in a sequence is called the *cycle time*, denoted by T. We consider a class of schedules where the cycle time for each server in the network is identical. However, as queues can be served multiple times in a sequence, it is possible for a server to perform its series of operations multiple times while another server completes a single series. For example, for sequence ({1,2},{2,4},{1,3},{3,4}) for the system depicted in Figure 5.1, server 1 serves queues 1 and 4 twice in a cycle (possibly with different service periods), while server 2 serves queues 2 and 3 once.

For operational or safety reasons, the cycle time can be restricted. For instance, minimal and maximal cycle times, respectively $T^{\min}$ and $T^{\max}$, can be taken into account

$$T^{\min} \leq T \leq T^{\max}. \tag{5.3}$$

The duration of a service period for queue n in group g_i is denoted by $\tau_{i,n}$.

A service schedule for sequence $\hat{s} = (\{1,2\},\{1,3\},\{1,2\},\{3,4\})$ is presented in Figure 5.2. For both servers, the setup and service periods of each group are depicted during a single cycle. It can be seen that queue 1 is served in three consecutive groups, hence no setups are required in the second and third group. Furthermore, queues 2 and 3 both have two service periods in this sequence. For this sequence, the service period of the second group α_2 equals the service period of queue 1 in that group $\tau_{2,1}$, as no setup period is required. This service period also equals the sum of setup and service periods of queue 3 in that group, hence $\alpha_2 = \tau_{2,1} = \sigma_{2,3} + \tau_{2,3}$.

Note that the start of serving a queue in a group depends on the setup period. Therefore, the starting times of service of the queues in a group can be different. However,

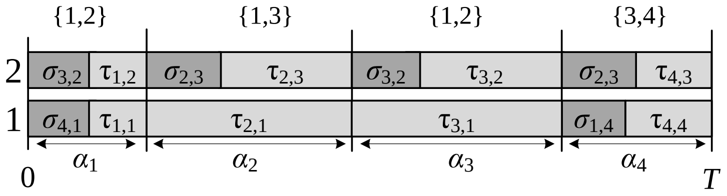


Figure 5.2: Service schedule for sequence $\hat{s}$ for the system depicted in Figure 5.1.

the service periods do end at the same time, for all queues in the group, provided that the server is not contained in the successive group, e.g., at the end of group $\{3,4\}$ in the example. Hence, switching groups indicates a synchronized switch between serving queues, i.e., after the switch the servers start serving the queues of the next group, with a possible setup time first. However, since queues can be contained in multiple groups, a switch between groups does not imply that service ends for all queues served before the switch.

Also, for operational or safety purposes, minimal or maximal duration of service periods can be required, denoted by

$$\tau_n^{\min} \leq \sum_{i\in g} \tau_{i,n} \leq \tau_n^{\max} \qquad \forall g \in P_{s,n}, \quad n = 1,2,\ldots,N. \tag{5.4}$$

Below, the problem of determining the optimal periodic schedule is formulated.

Problem Formulation

The optimization problem considered in this chapter can be summarized as follows: *For a multiclass fluid flow queueing network, derive the optimal periodic feasible service schedule*

This problem is solved in two steps. First, all feasible sequences are generated. This has been explained in detail in Section 4.4 and summarized below. Second, the optimal service periods are derived for each feasible sequence, presented in Section 5.3.

5.2 Sequence generation

With the approach presented in Section 4.4, the feasible sequences are derived in three steps. First, all feasible groups are derived, i.e., groups of queues that can be served simultaneously. Since we consider networks of servers, a group consists of $\geq S$ queues. Groups with $< S$ queues are not considered since at least one server is idle, which is not desired. Also, a group with $> S$ queues is infeasible, as each server is allowed to serve only one queue at a time. Therefore, the total number of feasible groups is given by $\Pi_{j=1}^{S}|Z(j)|$. For the network presented in Figure 5.1, the set of all feasible groups is given by

$$\mathscr{G} = \{\{1,2\},\{1,3\},\{2,4\},\{3,4\}\}.$$

Second, all feasible combinations of these groups are derived. Each combination satisfies (5.1) and each queue is served at least once in a sequence. Last, all permutations of these combinations of groups are derived. The resulting sequences that satisfy the maximal number of service periods for a queue (5.2), provide all feasible sequences. Note that increasing the bounds G and Θ results in (many) more feasible sequences.

5.3 Sequence optimization

For each feasible sequence, we derive the optimal service schedule. From these schedules, the schedule with the best performance renders the optimal periodic schedule. The service schedule of a feasible sequence is optimized using linear programming (LP) for minimizing the cycle time or quadratic programming (QP) for minimizing the weighted average wip level or weighted average flow time. The decision variables for sequence s are the service period durations $\tau_{i,n}$, for all $g_i \in s$ and $n \in g_i$.

There exist a variety of performance criteria. Similar to Chapter 4, we consider minimizing the cycle time, weighted average work in progress (wip) level (amount of vehicles regarding traffic intersections), weighted average flow (waiting) time, or a linear combination of these. For completeness, the different performance criteria, along with the optimization problem, are briefly discussed below.

5.3.1 Cycle time

The minimal cycle time is the minimal time that is required for a feasible schedule, i.e., to satisfy all constraints. The minimal required cycle time does not necessarily imply that the network is working at full capacity for the queues with the highest load in each group. Due to both constraints (5.3) and (5.4), the network might still have additional capacity at the minimal cycle time (which may be necessary to handle fluctuations in arrival rates, that typically occur in practice). Moreover, we use the minimal cycle time of a schedule as a lower bound for the cycle time to reduce the calculation effort of optimizing the weighted average wip level and flow time, as shown in Section 5.3.2. The cycle time for sequence s can be expressed as

$$T = \sum_{i=1}^{|s|} \alpha_i. \tag{5.5}$$

Given this criterion, the problem is to find the optimal duration of service and idle periods for each queue in a given sequence. Using $\tau_{i,n}$, the service periods for queue $n \in g_i$ in group $g_i \in s$, as decision variables, the cycle time T (5.5) can be expressed as the sum of service and idle periods of a queue. The idle periods of queue n are a combination of setup periods and service periods of queues in groups which do not contain queue n. For example, considering schedule $\hat{s}$, $T = 2\sigma_{3,2} + 2\sigma_{2,3} + \tau_{1,2} + \tau_{2,3} + \tau_{3,2} + \tau_{4,3}$. Hence, the cycle time is a linear combination of several service periods of queues and setup periods.

Next, the constraints are introduced, which depend on both the route of each flow through the network and the sequence. To do so, we first consider service of successive queues. Since a network can contain multiple servers, a group can consists of two or more successive queues, i.e., following the same flow, served by different servers. In the remainder of this chapter we use the following distinction between active and inactive queues. A queue n is *active* in group g_i, if it is served in that group, i.e., $n \in g_i$, otherwise, queue n is inactive in group g_i. Consider, for instance, queue n and upstream queue $p = n - 1$, which are both part of group g_i and therefore both active in group g_i. Here, during service of group g_i, queue n receives a piecewise-constant rate of fluid from the upstream queue p, i.e., at rates 0, μ_p, and λ_p, respectively, which complicates the formulation of the constraints for the service periods in comparison to the situation in which all queues receive a constant inflow (presented in Chapter 4). In order to derive constraints for the service periods, we need to divide the setup and service period of queue n in group g_i into three phases,

$$\alpha_i = \tau_{i,n}^0 + \tau_{i,n}^1 + \tau_{i,n}^2, \quad \forall n \in g_i, \quad \forall g_i \in s,$$

where the phases are defined by:

$\tau_{i,n}^0$: Setup period.

$\tau_{i,n}^1$: Service period of fluid that is in queue n at the start of serving group g_i, i.e., fluid that has arrived *before* the start of serving group g_i.

$\tau_{i,n}^2$: Service period of fluid that has arrived *after* the start of serving group g_i.

Note that the phrase “start of serving group g_i" indicates the start of (setting up to) serving the queues in group g_i. Service period constraints for queue n depend on the number of active upstream queues. We present the constraints for a queue without (active) upstream queue, with a single active upstream queue and with multiple active upstream queues below.

No (active) upstream queue

Consider active queue n, in group g_i, without a predecessor, i.e., $\lambda_n(t) = \lambda_n > 0$, depicted in Figure 5.3 along with the successive service rates and phases. Let $\tau_{i,n}^{r,k}$ denote the service period of queue n at rate r that occurs for the k-th time in group g_i. It can be seen that rate μ_n is present twice and the service durations at these rates are labeled by $\tau_{i,n}^{\mu_n,1}$ and $\tau_{i,n}^{\mu_n,2}$, respectively.

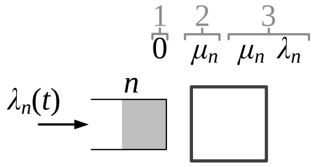


Figure 5.3: Arrival and service rates of queue n during service of group g_i. Fluid arrives from an external source with constant rate $\lambda_n(t) = \lambda_n > 0$ or queue n has no active upstream queue, so $\lambda_n(t) = 0$.

The first service rate, 0, of each queue is the rate during the setup period, i.e., phase 1. The second rate, μ_n, is the service rate during phase 2. The remaining rates indicate the service rates during the final phase, i.e., serving arrived fluid after the start of service of group g_i. This phase consists of a duration $\tau_{i,n}^{\mu_n,2}$ serving at maximal rate, and thereby the server is able to serve the fluid that has arrived since the start of serving this group, and duration $\tau_{i,n}^{\lambda_n,1}$, which denotes the service period of serving at arrival rate λ_n, i.e., when queue n has been emptied. The evolution of queue content x_n during service of group g_i is presented in Figure 5.4. The content of queue n at the start of serving group g_i is denoted by $x_{i,n}$. It can be seen that the queue content increases with rate λ_n while the server sets up to serve queue n (phase 1). Next, the queue is served at maximal rate for a duration of $\tau_{i,n}^{\mu_n,1}$ (phase 2) and $\tau_{i,n}^{\mu_n,2}$ (phase 3), i.e., a slope of $\lambda_n - \mu_n$ in the figure. Finally, if the queue has been emptied, arrivals are served at arrival rate, i.e., the queue remains empty. Note that in this figure the queue is completely emptied, a switch to another group is also possible when the queue has not been emptied yet. In that case, $\tau_{i,n}^{\lambda_n,1} = 0$.

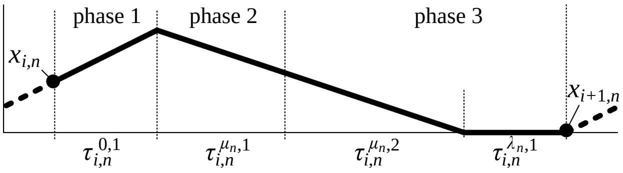


Figure 5.4: Evolution of queue content x_n during service in group g_i.

Next, bounds on the service periods of the queues are presented. The initial queue content $x_{i,n}$ is served at maximal rate for a duration of $\tau_{i,n}^{\mu_n,1}$ (phase 2). The maximal amount of fluid that can be served during this phase equals $x_{i,n}$, rendering an upper bound for the duration of this phase:

$$\mu_n \tau_{i,n}^{\mu_n,1} \leq x_{i,n}. \tag{5.6a}$$

Moreover, queue content x_n is non-negative, resulting in an upper bound on the service periods at maximal rate:

$$(\mu_n - \lambda_n)(\tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}) \leq x_{i,n} + \lambda_n \tau_{i,n}^{0,1}. \tag{5.6b}$$

Furthermore, the queue content at the end of serving group g_i can be derived from arrival and service periods:

$$x_{i+1,n} = x_{i,n} + \lambda_n \tau_{i,n}^{0,1} + (\lambda_n - \mu_n)(\tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}). \tag{5.6c}$$

For periodic behavior, the amount of fluid processed in a single cycle equals the amount of fluid arrived in a single cycle, i.e., if queue n is only served in group g_i:

$$\lambda_n T = \mu_n(\tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}) + \lambda_n \tau_{i,n}^{\lambda_n,1}. \tag{5.7}$$

For queue n with inactive predecessor, i.e., $\lambda_n = 0$, constraints (5.6) are identical and the queue evolution is similar as depicted in Figure 5.4, where the queue content is constant during phase 1. For (5.7), the arrival rate λ_n must be replaced with the arrival rate at the first queue of the flow.

Single active upstream queue

Here, we consider a queue with a 'single' active upstream queue, i.e., the upstream queue has no active upstream queue. For example, consider queue n in Figure 5.5 that is served in group g_i with active upstream queue p, i.e., $p,n \in m$. Queue p is the first queue on the route and therefore receives fluid from an external source ($\lambda_p(t) = \lambda_p > 0$).

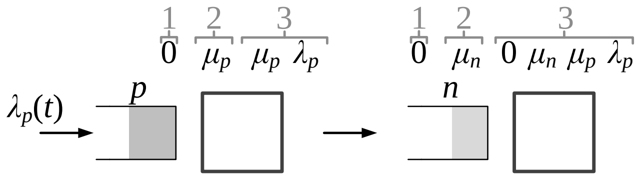


Figure 5.5: Arrival and service rates of two successive queues p and n during service of group g_i. Fluid arrives from an external source with constant rate $\lambda_p(t) = \lambda_p > 0$ or queue p has no active upstream queue, so $\lambda_p(t) = 0$.

During service of queue n, the server can serve at six successive rates, as presented in Figure 5.5. Compared to the queue without (active) predecessor, the third phase differs. For queue n, phase 3 consists of an idle period $\tau_{i,n}^{0,2}$, which occurs if the setup period in queue p takes longer than both the setup period for queue n and clearing the fluid in queue n, i.e., if $\tau_{i,p}^{0,1} > \tau_{i,n}^{0,1} + \tau_{i,n}^{\mu_n,1}$. Note that transportation delays are not taken into account, i.e., fluid is, while served, immediately transported to the successive queue. Next, for a duration of $\tau_{i,n}^{\mu_n,2}$ the fluid is served that has arrived since the start of serving this group. Once queue n is empty and queue p is not, which can only occur if $\mu_n > \mu_p$, queue n is served at arrival rate μ_p for a duration of $\tau_{i,n}^{\mu_p,2}$. Finally, when both queues are empty, queue n is served at arrival rate λ_p for a duration of $\tau_{i,n}^{\lambda_p,1}$. For clarity, we list the service periods below:

$\tau_{i,n}^{0,1}$: Setup period.

$\tau_{i,n}^{\mu,1}$: Time serving the fluid that is initially in the queue, i.e., $x_{i,n}$, at maximal rate.

$\tau_{i,n}^{0,2}$: Idling period. This idling period occurs when the initial fluid content of queue n is served and queue p has not served any fluid yet due to the setup, i.e., $\tau_{i,n}^{0,1} + \tau_{i,n}^{\mu,1} < \tau_{i,p}^{0,1}$.

$\tau_{i,n}^{\mu,2}$: Time serving fluid received from queue p since the start of serving this group at rate μ_n.

$\tau_{i,n}^{\mu_p,1}$: Time serving arriving fluid at rate μ_p, when $\mu_p < \mu_n$ and $x_n = 0$.

$\tau_{i,n}^{\lambda_p,1}$: Time serving arriving fluid at rate λ_p, i.e., the arrival rate of the upstream queue.

These service period durations are bounded by the service periods of upstream queue p and initial queue content $x_{i,n}$, presented below. Since queues are served at maximal rate, the time serving queue n at rate λ_p (that is, arrival rate of queue p) can not exceed the time serving queue p at arrival rate:

$$\tau_{i,n}^{\lambda_p,1} \leq \tau_{i,p}^{\lambda_p,1}. \tag{5.8a}$$

The time serving queue n, when empty, at rate μ_p can not exceed the time serving queue p at rate μ_p in phase 3, i.e.,

$$\tau_{i,n}^{\mu_p,1} \leq \tau_{i,p}^{\mu_p,1} + \tau_{i,p}^{\mu_p,2}. \tag{5.8b}$$

Total amount of fluid served in queue n in phase 3 can not exceed the total amount of fluid served in queue p:

$$\mu_n \tau_{i,n}^{\mu_n,2} + \mu_p \tau_{i,n}^{\mu_p,1} + \lambda_p \tau_{i,n}^{\lambda_p,1} \leq \mu_p(\tau_{i,p}^{\mu_p,1} + \tau_{i,p}^{\mu_p,2}) + \lambda_p \tau_{i,p}^{\lambda_p,1}. \tag{5.8c}$$

In phase 2, the amount of fluid served in queue n can not exceed the initial amount of fluid:

$$\mu_n \tau_{i,n}^{\mu_n,1} \leq x_{i,n}. \tag{5.8d}$$

The server can start serving fluid from queue n that has arrived from queue p (phase 3) whenever queue p is served, i.e., when the setup period for server p has expired:

$$\tau_{i,p}^{0,1} \leq \tau_{i,n}^{0,1} + \tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{0,2}. \tag{5.8e}$$

The queue content $x_{i+1,n}$ at the end of serving group g_i can be defined as:

$$\begin{aligned} x_{i+1,n} = & x_{i,n} + \mu_p(\tau_{i,p}^{\mu_p,1} + \tau_{i,p}^{\mu_p,2}) + \lambda_p \tau_{i,p}^{\lambda_p,1} - \mu_n(\tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}) \\ & - \mu_p \tau_{i,n}^{\mu_p,1} - \lambda_p \tau_{i,n}^{\lambda_p,1}. \end{aligned} \tag{5.8f}$$

For periodic behavior, the amount of fluid processed in a single cycle equals the amount of fluid arrived in a single cycle, i.e., if queue n is only served in group g_i:

$$\lambda_p T = \mu_n(\tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}) + \mu_p \tau_{i,n}^{\mu_p,1} + \lambda_p \tau_{i,n}^{\lambda_p,1}. \tag{5.9}$$

If queue p is not the first queue on the route and the upstream queue $p-1$ is inactive, i.e., $\lambda_p = 0$, constraints (5.8) are identical. For (5.9), the arrival rate λ_p must be replaced with the arrival rate at the first queue of the flow.

Multiple active upstream queues
For a server with multiple active upstream queues, constraints on service durations and queue contents can be derived similarly, but these are omitted for ease of exposition. Note that, if q is the number of active upstream queues, without intermediate inactive queues, then the number of successive service rates is given by $3+\sum_{i=1}^{q}(2+i)$.

The presented service period constraints (5.6)-(5.9), along with (5.3) and (5.4), complete the optimization problem. It can be seen that the constraints are all linear with respect to the service periods. Combined with a linear objective function, we conclude that minimizing the cycle times is a LP problem.

5.3.2 Weighted average wip level

Another criterion is the weighted time averaged wip (work in progress) level, or total queue contents in the network, given by:

$$J_w = \sum_{n=1}^{N} c_n W_n, \tag{5.10}$$

with c_n the weight factor and W_n the time average wip of queue n, given by

$$W_n = \frac{1}{T} w_n = \frac{1}{T} \int_0^T x_n(\tau) d\tau, \tag{5.11}$$

where w_n is the total wip during T. The average length of a single queue can be described by the duration of the service and idle periods of the corresponding server during a cycle, as shown below. Note that W_n in (5.11), by dividing by T, results in a non-linear function of the service periods $\tau_{i,n}$. Therefore, to use quadratic programming, the cycle time is assumed constant. Then, for a range of cycle times, the QP problems are solved and the best solution is chosen. For queue n in Figure 5.3, without (active) upstream queue, the total wip during service of group g_i, denoted by $w_{i,n}$, equals the area below the graph depicted in Figure 5.4, can be expressed by:

$$\begin{aligned} w_{i,n} =& x_{i,n}(\tau_{i,n}^{0,1} + \tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}) + \lambda_n \tau_{i,n}^{0,1}(\tfrac{1}{2}\tau_{i,n}^{0,1} + \tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}) - \\ & (\mu_n - \lambda_n)[\tau_{i,n}^{\mu_n,1}(\tfrac{1}{2}\tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{\mu_n,2}) + \tfrac{1}{2}(\tau_{i,n}^{\mu_n,2})^2]. \end{aligned} \tag{5.12}$$

Using the service periods of the active queues, wip levels for the inactive queues can be derived similarly. Then, the time averaged wip of queue n for sequence s, as denoted in (5.11), follows from

$$W_n = \frac{1}{T} \sum_{i=1}^{|s|} w_{i,n}. \tag{5.13}$$

For queue n with a single active upstream queue p, as depicted in Figure 5.5, it is complicated to provide an exact description of the evolution of queue n during the active period, due to the duration of service at different rates at both queues. However, by regarding both queues as a single *aggregate* queue, it has a constant input and known output, i.e., the input of the aggregated queues p and n is the constant input λ_p and the output is given by the service rates and service durations of queue n (which are linked to the service durations at queue p by (5.8a)-(5.8e)). Note that instant transportation between queues n and p is assumed. The evolution of the contents of the aggregate queue $\hat{x}_n = x_p + x_n$ during service in group g_i is depicted in Figure 5.6. Note that the indicated phases in the figure correspond to the phases of serving queue n.

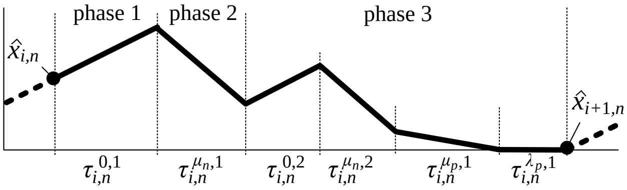


Figure 5.6: Evolution of $\hat{x}_n$ during service in group g_i.

The total wip $\hat{w}_n$ of the aggregate queue $\hat{x}_n$ during service of group g_i is expressed by:

$$\begin{aligned}\hat{w}_{i,n} =& x_{i,n}[\alpha_m - \tau_{i,n}^{\lambda_p,1}] + \lambda_p \tau_{i,n}^{0,1}[\alpha_m - \tau_{i,n}^{\lambda_p,1} - \tfrac{1}{2}\tau_{i,n}^{0,1}] + \\ & (\lambda_p - \mu_n)\tau_{i,n}^{\mu_n,1}[\tfrac{1}{2}\tau_{i,n}^{\mu_n,1} + \tau_{i,n}^{0,2} + \tau_{i,n}^{\mu_n,2} + \tau_{i,n}^{\tilde{\mu}_{(p,n)},1}] + \lambda_p \tau_{i,n}^{0,2}[\tfrac{1}{2}\tau_{i,n}^{0,2} + \tau_{i,n}^{\mu_n,2} + \tau_{i,n}^{\tilde{\mu}_{(p,n)},1}] + \\ & (\lambda_p - \mu_n)\tau_{i,n}^{\mu_n,2}[\tfrac{1}{2}\tau_{i,n}^{\mu_n,2} + \tau_{i,n}^{\tilde{\mu}_{(p,n)},1}] + (\lambda_p - \mu_{(p,n)})\tau_{i,n}^{\mu_p,1}[\tfrac{1}{2}\tau_{i,n}^{\mu_p,1}].\end{aligned}$$

With $w_{i,p}$ from (5.12), the total wip of queue n during this period can be derived by

$$w_{i,n} = \hat{w}_{i,n} - w_{p,m}.$$

Then, the time average wip of queue n is derived by (5.13). It can be seen that the objective functions are quadratic with respect to the service periods, provided that the cycle time T is *constant*. Combined with the constraints (5.3), (5.4), (5.6) and (5.8) this results in a QP problem.

To derive the optimum for a given sequence using QP, the solution of the QP problem must be derived for all cycle times in the range (5.3). However, by deriving the minimal required cycle time T^*, using LP as shown in Section 5.3.1, the optimum is found by minimizing the objective function over the range $\max(T^{\min}, T^*) \leq T \leq T^{\max}$. An efficient algorithm, e.g., the bisection method, can be used to locate the optimum. Comparing the optima for each feasible sequence results in the optimal periodic schedule.

The QP problem for the system depicted in Figure 5.1 and a given feasible sequence, with minimizing weighted average wip level as criterion, is presented, as an example, in Section 5.4.1.

5.3.3 Weighted average flow time

The weighted average flow time is also a criterion that we can consider. The flow time is the duration between entering and leaving the network. From Little's law, the following relation holds between the average flow time φ_n and the average wip W_n:

$$\varphi_n = \frac{W_n}{\lambda_n}.$$

Therefore, the weighted average flow time J_φ is given by:

$$J_\varphi = \sum_{n=1}^{N} \frac{c_n W_n}{\lambda_n}.$$

So, in fact, J_w and J_φ can both be considered as a weighted average wip level.

5.3.4 Linear combination of criteria

Instead of optimizing a single performance criterion, these can also be linearly combined. One can, for instance, minimize both weighted wip and flow time. By adjusting costs, emphasis can be put on a specific criterion.

Remark 5.3.1. *Note that setup costs are not taken into account in these criteria. Since, for a given sequence, the number of switches is fixed, and hence the setup costs are fixed. Therefore, they do not play a role in the optimization process for a single sequence, but can be taken into account when comparing sequences.*

5.4 Illustrations

This section provides some examples of optimal service schedules for multi-queue fluid flow networks. Optimal schedules for the Kumar-Seidman network and a three-server network are presented.

5.4.1 Kumar-Seidman network

In [68] the problem of minimizing the weighted average wip is considered for the Kumar-Seidman network. Using the method presented in this chapter, we also derive the optimal periodic behavior for this network. As an example, a part of the QP formulation for sequence $(\{1,2\},\{2,4\},\{3,4\})$ is presented below. For readability purposes, only the constraints involving queue 2 are presented. The constraints of the other queues are similar, and therefore omitted in this overview. The objective function is given by

$$\min_{\tau_1,\tau_2,\tau_3,\tau_4,\tau_5} \quad \frac{1}{T}(c_1 W_1 + c_2 W_2 + c_3 W_3 + c_4 W_4)$$

with

$$
\begin{aligned}
W_2 =& \tfrac{1}{T}(w_{1,2}+w_{2,2}+w_{3,2}),\\
w_{1,2} =& x_{1,2}[\alpha_1 - \tau_{1,2}^{\lambda_p,1}] + \lambda_p \tau_{1,2}^{0,1}[\alpha_1 - \tau_{1,2}^{\lambda_p,1} - \tfrac{1}{2}\tau_{1,2}^{0,1}] +\\
& (\lambda_p - \mu_2)\tau_{1,2}^{\mu_2,1}[\tfrac{1}{2}\tau_{1,2}^{\mu_2,1} + \tau_{1,2}^{0,2} + \tau_{1,2}^{\mu_2,2} + \tau_{1,2}^{\tilde{\mu}_{(p,2)},1}] + \lambda_p \tau_{1,2}^{0,2}[\tfrac{1}{2}\tau_{1,2}^{0,2} + \tau_{1,2}^{\mu_2,2} + \tau_{1,2}^{\tilde{\mu}_{(p,2)},1}] +\\
& (\lambda_p - \mu_2)\tau_{1,2}^{\mu_2,2}[\tfrac{1}{2}\tau_{1,2}^{\mu_2,2} + \tau_{1,2}^{\tilde{\mu}_{(p,2)},1}] + (\lambda_p - \mu_{(p,2)})\tau_{1,2}^{\mu_p,1}[\tfrac{1}{2}\tau_{1,2}^{\mu_p,1}]\\
& - x_{1,1}(\tau_{1,1}^{0,1} + \tau_{1,1}^{\mu_1,1} + \tau_{1,1}^{\mu_1,2}) + \lambda_1 \tau_{1,1}^{0,1}(\tfrac{1}{2}\tau_{1,1}^{0,1} + \tau_{1,1}^{\mu_1,1} + \tau_{1,1}^{\mu_1,2}) -\\
& (\mu_1 - \lambda_1)[\tau_{1,1}^{\mu_1,1}(\tfrac{1}{2}\tau_{1,1}^{\mu_1,1} + \tau_{1,1}^{\mu_1,2}) + \tfrac{1}{2}(\tau_{1,1}^{\mu_1,2})^2].\\
w_{2,2} =& x_{2,2}(\tau_{2,2}^{0,1} + \tau_{2,2}^{\mu_2,1} + \tau_{2,2}^{\mu_1,2}) + \lambda_1 \tau_{2,2}^{0,1}(\tfrac{1}{2}\tau_{2,2}^{0,1} + \tau_{2,2}^{\mu_2,1} + \tau_{2,2}^{\mu_2,2}) -\\
& (\mu_1 - \lambda_1)[\tau_{2,2}^{\mu_2,1}(\tfrac{1}{2}\tau_{2,2}^{\mu_2,1} + \tau_{2,2}^{\mu_1,2}) + \tfrac{1}{2}(\tau_{2,2}^{\mu_2,2})^2],\\
w_{3,2} =& x_{3,2}\alpha_3,\\
x_{2,2} =& x_{1,2} + \mu_1(\tau_{1,1}^{\mu_1,1} + \tau_{1,1}^{\mu_1,2}) + \lambda_1 \tau_{1,1}^{\lambda_1,1} - \mu_2(\tau_{1,2}^{\mu_2,1} + \tau_{1,2}^{\mu_2,2}) - \mu_1 \tau_{1,2}^{\mu_1,1} - \lambda_1 \tau_{1,2}^{\lambda_1,1},\\
x_{3,2} =& x_{2,2} - \mu_2(\tau_{2,2}^{\mu_2,1} + \tau_{2,2}^{\mu_2,2}).
\end{aligned}
$$

The constraints, involving queue 2 only, are as follows:

$$
\begin{aligned}
\alpha_1 &= \sigma_{3,2} + \tau_{1,2}^{\mu_2,1} + \tau_{1,2}^{0,2} + \tau_{1,2}^{\mu_2,2} + \tau_{1,2}^{\mu_1,1} + \tau_{1,2}^{\lambda_1,1},\\
\alpha_2 &= \tau_{2,2}^{\mu_2,1} + \tau_{2,2}^{\mu_2,2} + \tau_{2,2}^{\lambda_2,1},\\
T &= \alpha_1 + \alpha_2 + \alpha_3,\\
\lambda_1 T &= \mu_2(\tau_{1,2}^{\mu_2,1} + \tau_{1,2}^{\mu_2,2} + \tau_{2,2}^{\mu_2,1} + \tau_{2,2}^{\mu_2,2}) + \mu_1 \tau_{1,2}^{\mu_1,1} + \lambda_1 \tau_{1,2}^{\lambda_1,1},
\end{aligned}
$$

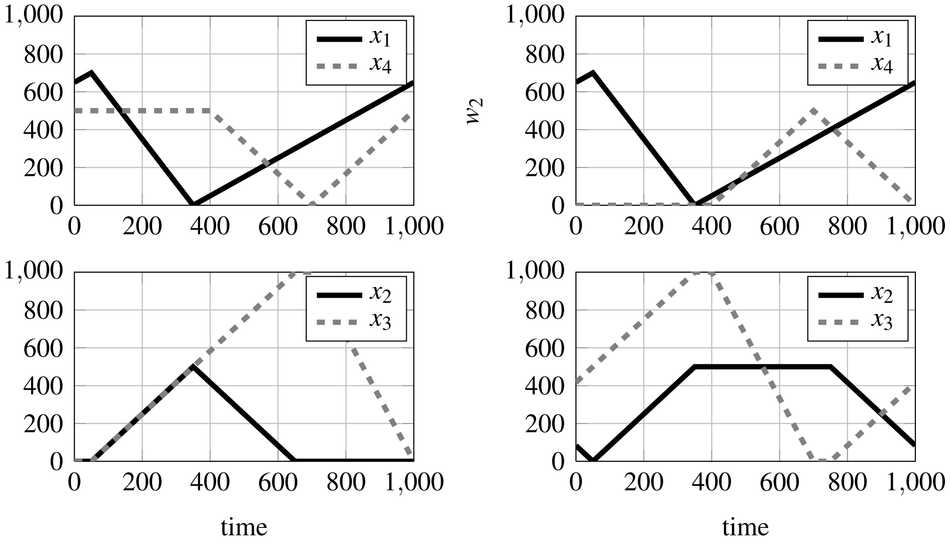
$$
\begin{aligned}
\tau_{1,2}^{\lambda_p,1} \leq& \tau_{1,p}^{\lambda_p,1},\\
\tau_{1,2}^{\mu_p,1} \leq& \tau_{1,p}^{\mu_p,1} + \tau_{1,p}^{\mu_p,2},\\
\mu_2 \tau_{1,2}^{\mu_2,2} + \mu_p \tau_{1,2}^{\mu_p,1} + \lambda_p \tau_{1,2}^{\lambda_p,1} \leq& \mu_p(\tau_{1,p}^{\mu_p,1} + \tau_{1,p}^{\mu_p,2}) + \lambda_p \tau_{1,p}^{\lambda_p,1},\\
\mu_2 \tau_{1,2}^{\mu_2,1} \leq& x_{1,2},\\
\tau_{1,1}^{0,1} \leq& \tau_{1,2}^{0,1} + \tau_{1,2}^{\mu_2,1} + \tau_{1,2}^{0,2},\\
\mu_2 \tau_{2,2}^{\mu_2,1} \leq& x_{2,2},\\
(\mu_2 - \lambda_2)(\tau_{2,2}^{\mu_2,1} + \tau_{2,2}^{\mu_2,2}) \leq& x_{2,2} + \lambda_2 \tau_{2,2}^{0,1}.
\end{aligned}
$$

The parameters considered in [68] are

$$
\begin{aligned}
\lambda_1 &= 1,\\
\sigma_{1,4} = \sigma_{4,1} = \sigma_{2,3} = \sigma_{3,2} &= 50,\\
\mu_1 = \mu_3 = \tfrac{1}{0.3},\ \mu_2 &= \mu_4 = \tfrac{1}{0.6}.
\end{aligned}
$$

The objective is to minimize the weighted average wip level (5.10). Moreover, equal costs are assumed for all queues, i.e., $c_1 = c_2 = c_3 = c_4 = 1$, and no setup costs are considered. For these parameters, the minimal cycle time $T^* = 1000$ time units, which follows from solving the LP problem presented in Section 5.3.1.

For $G = 3$, and therefore $\Theta = 1$, we find two sequences, $s_1^* = (\{1,2\},\{2,4\},\{3,4\})$ and $s_2^* = (\{1,2\},\{3,4\},\{2,4\})$, with $J_w^* = 1350$. Sequence s_1^* corresponds to the sequence in [68]. The results are presented in Figure 5.7. Both sequences consist of identical groups, but the ordering of the groups is different. Durations of serving groups are also the same, $\alpha_{\{1,2\}} = 350$, $\alpha_{\{2,4\}} = 300$ and $\alpha_{\{3,4\}} = 350$. It can be seen that the queue contents of queues 1 and 3 are identical for both sequences, as these queues are served only in a single group. Furthermore, the wip levels of queues 2 and 4 are identical to the the wip levels of queues 4 and 2 of the other sequence. For the network with emphasis on the first queues of the network, i.e., $c_2 > c_4$, sequence s_1^* is optimal, and s_2^* vice versa.



(a) Sequence $(\{1,2\},\{2,4\},\{3,4\})$. **(b)** Sequence $(\{1,2\},\{3,4\},\{2,4\})$.

Figure 5.7: Optimal sequences, $G = 3$. Sequence $(\{1,2\},\{2,4\},\{3,4\})$ (left) and sequence $(\{1,2\},\{3,4\},\{2,4\})$ (right).

5.4.2 Three server network

As final example we introduce a network with three servers and two fluid flows, see Figure 5.8, and minimize the weighted average wip. One flow visits servers 3, 2 and 1, while the other flow visits servers 1, 2, 3 and 1 again. Arrival rates and costs are indicated in Figure 5.8. The service rate for all queues is 10. For this system, the total number of groups is 12 and the minimal number of groups in a sequence is 3, as server 1 has to process all queues at least once.

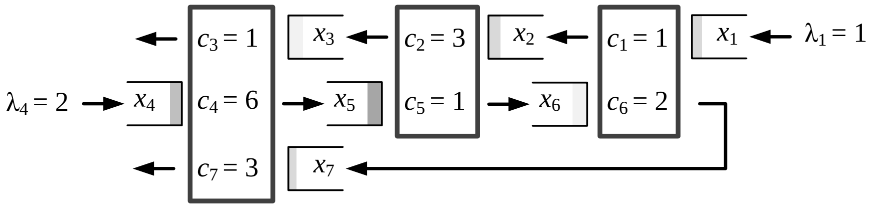


Figure 5.8: Three server network with two fluid flows.

For $G = 3$, and therefore $\Theta = 1$, there exist 72 feasible sequences and the optimal sequence is given by $s_1^* = (\{1,2,3\},\{1,4,5\},\{2,6,7\})$ with $J_w^* = 140.92$. For $G = 4$ and $\Theta = 1$, 720 feasible sequences exist. The optimal solution $s_2^* = (\{1,2,4\},\{5,6,7\},\{3,5,6\},\{1,3,5\})$ has a 0.5% ($J_w^* = 140.20$) better performance. For $G = 4$ and $\Theta = 2$, the number of feasible sequences grows to 1656. Then, the optimal solution is given by $s_3^* = (\{1,2,3\},\{2,4,6\},\{5,6,7\},\{2,4,6\})$ and has a 21.9% ($J_w^* = 115.85$) better performance! In this schedule, queue 4, which has the highest load, is served twice and is the main contributor to the performance improvement. Note that the number of feasible sequences quickly grows by relaxing the bounds on number of groups and total number of service periods per queue. Since each sequence is optimized separately, the computational efforts increase enormously. Therefore, the presented approach seems to be suitable for moderate networks of switching servers.

5.5 Summary

This chapter elaborates on the results of Chapter 4 and presents a method to derive an optimal periodic service schedule for a fluid flow queueing network with switching servers. Without assuming a policy a priory, we start with an objective and derive the optimal periodic service schedule with synchronized switches within the formulated constraints.

Similar to Chapters 3 and 4, the problem is tackled in two consecutive steps. First, all feasible sequences are derived. Second, for all feasible sequences, the optimal schedule, i.e., service period durations, are derived. These schedules are calculated separately with respect to possible restrictions on cycle time, service periods and maximal queue content. From these solutions, the best one is selected, which is the optimal periodic schedule for that network, given the restrictions on total number of groups in a sequence and maximal number of service periods for a queue. We show that the problem of finding the optimal service periods, given a sequence, can be formulated as a QP or LP. This creates flexibility in setting the constraints and objectives. Unlike aforementioned chapters, non-constant arrival patterns have been investigated in this chapter. These non-constant arrival patters occur as successive queues are served at the same time. This complicates the derivation of the queue contents, but it is possible by dividing the service periods into multiple phases.

By relaxing the bounds on the total number of groups in a sequence and the maximal number of service periods for a queue, the total number of feasible sequences increases rapidly, and thereby the required computational effort. A suggestion to decrease the computational effort is to regard a smaller set of feasible sequence, instead of all feasible sequences. One could for instance dismiss families of sequences beforehand or investigate the use of branch-and-bound techniques to derive the optimal schedule. Also, by adding (operational) restrictions the total number of feasible sequences, the computational effort can be decreased, e.g., after service of queue n by server j, server j is obliged to serve queue p.

Furthermore, instant transportation of fluid between queues has been assumed throughout the chapter, this may be a restrictive assumption in practice. By allowing transportation times, more applications can be described by the current model framework, such as, e.g., networks of signalized traffic intersections. To that end, for a certain two-server network with transportation times, we present a heuristic that provides a good service schedule in Chapter 9. First, the optimal service schedule for the system without transportation times is derived. Second, a phase delay is added to the schedule of one of the servers. Finally, this phase-delay is optimized.

Chapter 6

Transient behavior of a switching server

The periodic behavior of a single switching server has been investigated for two-queue servers in Chapter 3 and for multi-queue servers in Chapter 4. Starting from any feasible initial state, the transient behavior, i.e., trajectory to reach the desired periodic behavior, is presented in this chapter.

The transient optimization problem is that of steering the system towards the optimal steady-state trajectory at minimal costs. Machine failure in a manufacturing application or bus priorities in a signalized traffic intersection are two examples that can remove the system from the steady-state trajectory. We assume that deviations from the steady-state trajectory rarely occur, allowing the system to recover to the steady-state situation after each interruption, which is reasonable for systems such as traffic intersections or manufacturing applications.

In this chapter, the emphasis is on the two-queue switching server model of Chapter 3. For a two-queue server, a transient solution is defined as a trajectory in the $x_1 - x_2$ space that leads to the optimal steady-state trajectory in a *finite* amount of time. An *optimal* transient solution is a transient solution which minimizes the costs of reaching the optimal steady-state trajectory. A concise system description is provided in Section 6.1. In this chapter, the transient behavior is discussed for two classes of two-queue switching servers: servers without backlog and servers with backlog. For systems without backlog, the queue contents can be easily derived and the transient behavior is presented in Section 6.2. For systems with backlog, the derivation of the total amount of backlog during the transient period is more complex, and also the resulting transient behavior is more complicated, as discussed in Section 6.3. Section 6.4 presents transient behavior of a multi-queue switching server and a summary is provided in Section 6.5.

This chapter is partly based on [106, 108].

6.1 System description

In this chapter a system of two queues served by a single switching server is considered, see Figure 6.1. Fluid arrives at each queue $n = 1,2$ with arrival rate λ_n. The content of queue n at time t is denoted by $x_n(t)$. If backlog is allowed, the backlog level is denoted by $x_n^-(t) = \min(x_n(t), 0)$ and the inventory level by $x_n^+(t) = \max(x_n(t), 0)$. If the server serves queue n, the service rate is given by $r_n \in \{0, \lambda_n, \mu_n\}$. The workload of queue n is defined by $\rho_n = \frac{\lambda_n}{\mu_n}$. A setup period $\sigma_{i,j}$ is required for switching from queue i to queue j. The duration of a service (idle) period for queue n is nonnegative and is denoted by τ_n (τ_n^0). The idle period is divided into a part during which the server sets up to serve the queue and a part in which the server idles. The service period is also divided into two parts, i.e., a period of service at maximal rate τ_n^μ and a period of service at arrival rate τ_n^λ.

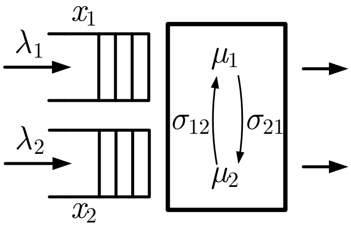


Figure 6.1: Two-queue switching server.

The state x of the system not only consist of queue levels x_1 and x_2, but also of the *remaining idle time* x_0 and group $g \in \mathscr{G}$. A group is a set of queues that can be served simultaneously and the set of all groups is denoted by $\mathscr{G}$. Given that both queues can not be served simultaneously, only two groups exist for the system in Figure 6.1, denoted by $g_1 = \{1\}$ and $g_2 = \{2\}$ ($\mathscr{G} = \{\{1\},\{2\}\}$). The state of the system is defined by

$$x(t) = \begin{bmatrix} x_0(t) & x_1(t) & x_2(t) & g(t) \end{bmatrix}^\top \in \mathbb{R}^3 \times \mathscr{G}. \tag{6.1}$$

The *cycle time* T is the sum of idle and service periods for all queues, i.e.,

$$T = \tau_1^0 + \tau_1^\mu + \tau_1^\lambda + \tau_2^0 + \tau_2^\mu + \tau_2^\lambda. \tag{6.2}$$

where the duration of the idle periods includes the setup period,

$$\sigma_{j,i} \leq \tau_i^0, \qquad i,j = 1,2, \quad j \neq i. \tag{6.3}$$

For stability of the system, i.e., the server is able to serve all arrivals in a cycle, the service periods must satisfy

$$\lambda_n T = \mu_n \tau_n^\mu + \lambda_n \tau_n^\lambda \qquad n = 1,2,\ldots,N.$$

For a more detailed description of this system, the reader is referred to Section 3.1.

6.2 System without backlog

In this section, backlog is not allowed for the considered two-queue switching server, i.e., $x_n(t) = x_n^+(t) \geq 0$, $n = 1,2$. Then, the transient costs are defined by

$$J_p = \liminf_{t\to\infty} \int_0^t c_1^+ x_1(\tau) + c_2^+ x_2(\tau) + s_{2,1} v_1(\tau) + s_{1,2} v_2(\tau) - J_w^* d\tau, \qquad (6.4)$$

where J_w^* are the optimal time average steady-state costs (see Section 3.2) and $v_n(t) = \frac{1}{\sigma_{j,n}}$, $j \neq n$ during a setup to queue n and $v_n(t) = 0$ otherwise. We denote by $x(0)$ the initial state immediately after moving away from the periodic solution, e.g., after the machine failure or bus priority. In order to reach the steady-state trajectory from every possible initial state in finite time, the steady-state trajectory requires a slow-mode, since serving at a lower rate, i.e., not at full capacity, provides the transient trajectory to 'catch up' with the steady-state trajectory.

Similar to the steady-state trajectories, idling of the server results in non-optimal behavior, as a counterexample can be derived without the idle periods that provides better performance. Therefore,

$$\tau_{i,c}^0 = \sigma_{j,i}, \qquad \text{for } i,j = 1,2,\, i \neq j, \quad c = 1,2,...,C, \qquad (6.5)$$

and if queue n is served, the service rate is given by

$$r_n(t) = \begin{cases} \mu_n & \text{if } x_n(t) > 0, \\ \lambda_n & \text{if } x_n(t) = 0. \end{cases}$$

In other words, if the queue is nonempty, service is at maximal rate, otherwise at arrival rate.

For a *fixed* number of cycles C, we present the transient optimization problem as a QP problem. A cycle, starting at group g, is defined as the series of operations until the end of service of the previous group (which is 2 for group 1 and 1 for group 2). Denote by $\tau_{n,c}$ the service period of queue n for the c-th cycle ($c \leq C$), consisting of the service period at maximal rate $\tau_{n,c}^\mu$ and the service period at arrival rate $\tau_{n,c}^\lambda$. In the remainder of this chapter we assume that the initial group is 1 and $x_0 = \sigma_{1,2}$, i.e., start setting up to serve queue 1, and derive the QP problem for this particular case. The QP problems for other initial situations can be derived similarly. Constraints for the transient problem are listed below. Minimal and maximal cycle time constraints are:

$$T^{\min} \leq T_c \leq T^{\max}, \qquad c = 1,2,...,C, \qquad (6.6a)$$

with $T_c = \tau_{1,c}^0 + \tau_{1,c}^\mu + \tau_{1,c}^\lambda + \tau_{2,c}^0 + \tau_{2,c}^\mu + \tau_{2,c}^\lambda$. Minimal and maximal service period durations are given by

$$\tau_n^{\min} \leq \tau_{n,c} \leq \tau_n^{\max}, \qquad \text{for } n = 1,2, \quad c = 1,2,...,C, \qquad (6.6b)$$

and minimal idle time is given by

$$\sigma_{j,n} \leq \tau_n^0, \qquad n,j = 1,2, \quad j \neq n, \quad c = 1,2,...,C. \tag{6.6c}$$

The bounds on the queue contents are given by:

$$x_n^{\min} \leq x_n(t) \leq x_n^{\max}, \qquad \text{for } n = 1,2. \tag{6.6d}$$

If the transient optimization problem is considered for an infinite number of cycles, the transient trajectory would remain on the steady-state trajectory once it is reached. However, due to the finite number of cycles considered in the QP problem, a termination effect occurs, e.g., elongating the cycle time and/or lowering the accumulated queue contents in the final cycle (or even earlier) can lower the costs. As an illustrative example, the queue levels of a transient trajectory with $C = 5$ is presented in Figure 6.2, along with the setup and service periods.

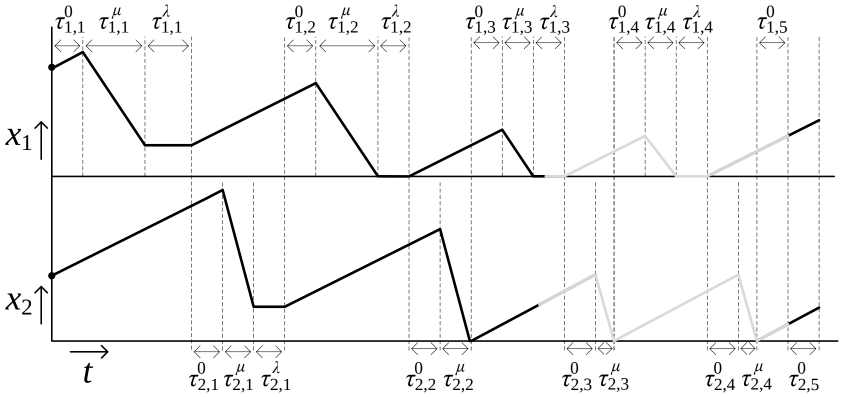


Figure 6.2: Queue levels during transient phase ($C = 5$), convergence to steady-state trajectory during $\tau_{1,3}^\lambda$.

Since the initial group is group 1 and $x_0(0) = \tau_{1,c}^0$, cycle c starts in group 1 and ends at the end of group 2, i.e., service at rate $\tau_{2,c}^\lambda$. The transient trajectory converges to the steady-state trajectory, presented in gray, during service of queue 1 at arrival rate in the third cycle, i.e., during $\tau_{1,3}^\lambda$. The trajectory remains on the steady-state trajectory until the fifth, and final, cycle. In the final cycle, service periods are zero and the steady-state trajectory is left, reflecting the termination effect. Note that the depicted trajectory is not the optimal transient trajectory, as for instance slow-modes $\tau_{1,1}^\lambda$ and $\tau_{2,1}^\lambda$ occur while the queues are non-empty. Let us denote by $x_{n,c}$ the content of queue n at the end of the c-th cycle:

$$x_{n,c+1} = x_{n,c} + \lambda_n T_c - \mu_n \tau_{n,c}^\mu, \quad n = 1,2, \quad c = 1,2,...,C, \tag{6.7}$$

where $x_{n,0} = x_n(0)$. To negate the termination effect, we enforce the final state of the final cycle C of the transient solution to be identical to the final state of the steady-state solution, i.e.,

$$x_{n,C} = x_n^*, \qquad n = 1,2, \tag{6.8}$$

where x_n^* is the content of queue n at the start/end of the optimal steady-state trajectory, i.e., the content at the start of the setup to serve queue 1. If (6.8) holds, the trajectory is defined as a *feasible* transient trajectory, otherwise the trajectory is *infeasible*. Note that in Figure 6.2 the trajectory is not feasible, as the final state deviates from the optimal steady-state trajectory.

The total queue contents $w_{n,c}$ of queue n for cycle c can be derived by:

$$\begin{aligned} w_{1,c} =& (x_{1,c-1} + \tfrac{1}{2}\lambda_1 \tau^0_{1,c})\tau^0_{1,c} + (x_{1,c-1} + \lambda_1 \tau^0_{1,c} - \tfrac{1}{2}\mu_1 \tau^\mu_{1,c})\tau^\mu_{1,c} + \\ &+ (x_{1,c-1} + \lambda_1 \tau^0_{1,c} - \mu_1 \tau^\mu_{1,c})\tau^\lambda_{1,c} + (x_{1,c-1} + \lambda_1 \tau^0_{1,c} + \\ &+ \tfrac{1}{2}\lambda_1(\tau^0_{2,c} + \tau^\mu_{2,c} + \tau^\lambda_{2,c}) - \mu_1 \tau^\mu_{1,c})(\tau^0_{2,c} + \tau^\mu_{2,c} + \tau^\lambda_{2,c}), \quad c = 1,2,\dots,C, \end{aligned} \tag{6.9a}$$

$$\begin{aligned} w_{2,c} =& (x_{2,c-1} + \tfrac{1}{2}\lambda_2(\tau^0_{1,c} + \tau^\mu_{1,c} + \tau^\lambda_{1,c} + \tau^0_{2,c}))(\tau^0_{1,c} + \tau^\mu_{1,c} + \tau^\lambda_{1,c} + \tau^0_{2,c}) + \\ &+ (x_{2,c-1} + \lambda_2(\tau^0_{1,c} + \tau^\mu_{1,c} + \tau^\lambda_{1,c} + \tau^0_{2,c}) - \tfrac{1}{2}\mu_2 \tau^\mu_{2,c})\tau^\mu_{2,c} + \\ &+ (x_{2,c-1} + \lambda_2(\tau^0_{1,c} + \tau^\mu_{1,c} + \tau^\lambda_{1,c} + \tau^0_{2,c}) - \mu_2 \tau^\mu_{2,c})\tau^\lambda_{2,c}, \quad c = 1,2,\dots,C. \end{aligned} \tag{6.9b}$$

Using (6.9), the transient costs (6.4), considering C cycles, can be written as

$$J_p(C) = C(s_{1,2} + s_{2,1}) + Q_p(C).$$

Here, $Q_p(C)$ is the solution to the quadratic programming problem, for C cycles, given by

$$Q_p(C) = \min_{\tau^\mu_{n,c}, \tau^\lambda_{n,c}} \sum_{n=1}^{2} \sum_{c=1}^{C} \left[c_i^+ w_{n,c} - J_w^*(\tau^0_{n,c} + \tau^\mu_{n,c} + \tau^\lambda_{n,c}) \right], \tag{6.10}$$

subject to constraints (6.6a)-(6.6c), (6.8) and

$$x_{1,c} \geq \lambda_1(\tau^0_{2,c} + \tau^\mu_{2,c} + \tau^\lambda_{2,c}), \qquad c = 1,2,\dots,C, \tag{6.11a}$$

$$x_{2,c} \geq 0, \qquad c = 1,2,\dots,C, \tag{6.11b}$$

$$x_{1,c-1} \leq x_1^{\max} - \lambda_1 \tau^0_{1,c}, \qquad c = 1,2,\dots,C, \tag{6.11c}$$

$$x_{2,c} \leq x_2^{\max} - (\mu_2 - \lambda_1)\tau^\mu_{2,c}, \qquad c = 1,2,\dots,C, \tag{6.11d}$$

where constraints (6.11a)-(6.11b) follow from $x_i(t) \geq 0$ and constraints (6.11c)-(6.11d) follow from (6.6d).

Given a system with the initial state outside the steady-state trajectory, the number of cycles required to derive the optimal transient trajectory is not easily determined, as shown by, for instance, the optimal trajectory in Figure 6.4. However, a lower bound on the number of cycles required for a feasible transient trajectory $C^{\min}$ can be determined by using a clearing policy, regarding the initial state and considering a system without capacity or service period constraints. For a system with capacity or service period constraints, this number of cycles is usually not enough to reach the steady-state trajectory. Starting from this lower bound, and by adding extra cycles, we solve the QP problem until a feasible transient trajectory is derived. Note that this transient trajectory is not necessarily the optimal trajectory, i.e., adding more cycles may lower the costs. Therefore, the number of cycles considered in the QP problem (6.10) is increased until the total costs required for the transient trajectory to reach the steady-state trajectory does no longer change, i.e., $J_p^*(C) = J_p^*(C+i)$, $\forall i > 0$. Then, adding more cycles does not result in a different transient trajectory, in the sense that it only adds steady-state cycles to the solution. Hence, we can then conclude that the transient solution is the optimal one.

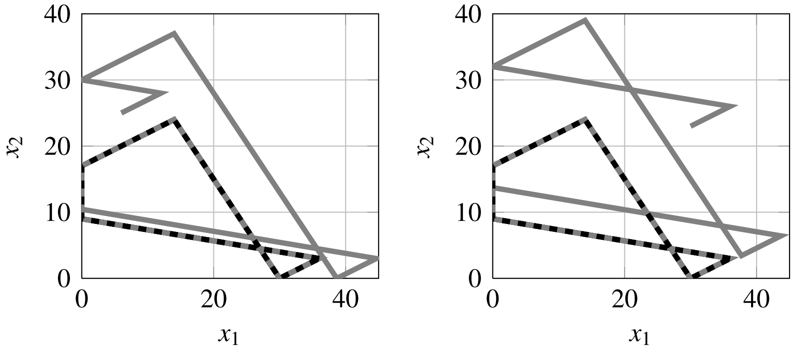
6.2.1 Illustrations

For the system with parameters

$$\begin{aligned} \lambda_1 &= 2, & \lambda_2 &= 1, \\ \mu_1 &= 8, & \mu_2 &= 4, \\ \sigma_{2,1} &= 3, & \sigma_{1,2} &= 7, \\ c_1^+ &= 8, & c_2^+ &= 1, \end{aligned} \tag{6.12}$$

and without constraints on queue length, cycle time and service periods, the optimal transient trajectory for initial state $x(0) = \begin{bmatrix} 3 & 6 & 25 & 1 \end{bmatrix}^\top$ is presented in Figure 6.3a by the solid line. The optimal periodic behavior is depicted by the intersected line. Note, that the initial group is 1, and that the steady-state trajectory is reached during the second cycle. It can be seen that for this initial state a clearing policy (until the steady-state trajectory is reached) yields the optimal performance. However, the optimal trajectory for the system with initial state $x(0) = \begin{bmatrix} 3 & 30 & 23 & 1 \end{bmatrix}^\top$, presented in Figure 6.3b, gives a different result. First, after the setup, queue 1 is emptied. Second, after the setup, queue 2 is served until a content of 3.43 is reached, then the system switches to serve queue 1. Note that queue 2 is not emptied. Next, queues 1 and 2 are both cleared before reaching the steady-state trajectory.

For the trajectory depicted in Figure 6.3b, it is clearly shown that a trade-off exists between a build-up of the much more expensive queue 1 and switching before emptying queue 2. This behavior is not present in symmetric systems, as a clearing policy is optimal for symmetric systems, see for instance [19, 73].



(a) $x(0) = \begin{bmatrix}3 & 6 & 25 & 1\end{bmatrix}^\top$. **(b)** $x(0) = \begin{bmatrix}3 & 30 & 23 & 1\end{bmatrix}^\top$.

Figure 6.3: Optimal transient trajectories with different initial states. In (a) the clearing policy is optimal, in (b) it is not.

Each optimal transient trajectory contains *switching points*. A switching point is the state $x = \begin{bmatrix}x_0 & x_1 & x_2 & g\end{bmatrix}^\top$ at which the system switches to serve the other queue, i.e., switching between groups $g = 1$ and $g = 2$ and between groups $g = 2$ and $g = 1$. Experimentally combining the switching points of optimal trajectories, i.e., solving the transient problem for a set of initial states and collecting the switching points, results in a *switching curve*. A switching curve characterizes the optimal transient structure for any given initial state, provided that the server works at maximal rate.

The (experimentally determined) switching curves for the system with parameters (6.12) are presented in Figure 6.4, along with a trajectory for initial state $x(0) = \begin{bmatrix}3 & 40 & 80 & 1\end{bmatrix}^\top$. The switching curve for a transition between groups $g = 1$ and $g = 2$ is given by the line starting from $x_1 = 0$ and $x_2 \geq 17$, where $(0, 17)$ is the switching point of the optimal steady-state trajectory. The switching curve for a transition between groups $g = 2$ and $g = 1$ is *discontinuous* with linear segments. These segments do not overlap, i.e., each initial state has a single optimal trajectory.

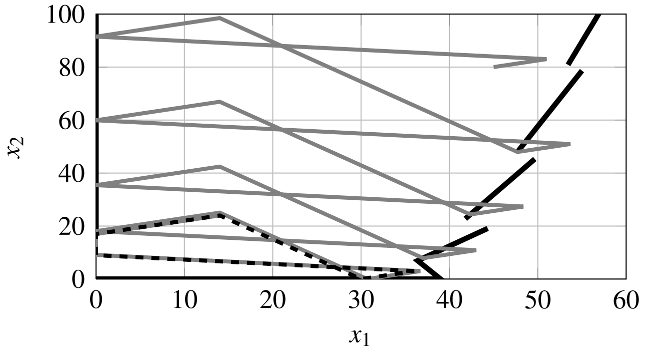
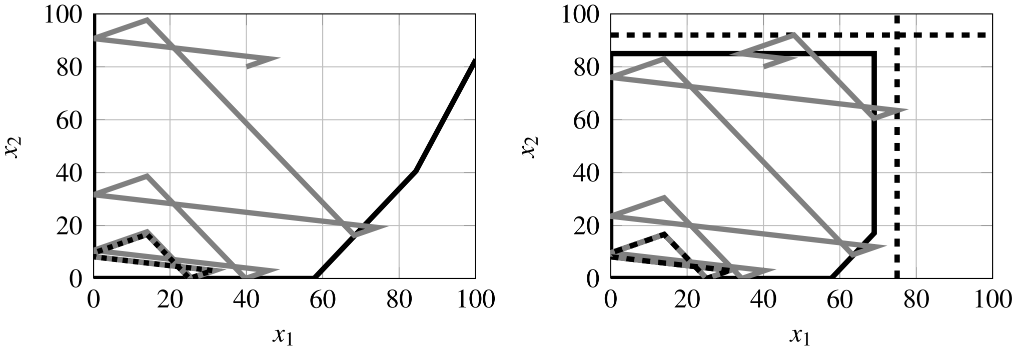


Figure 6.4: Discontinuous switching curves (black), for the system with parameters (6.12), and transient trajectory (gray) for $x(0) = [3 \;\; 40 \;\; 80 \;\; 1]^\top$.

For the system with parameters (6.12) and $c_1^+ = 2$, the switching curve is continuous, see Figure 6.5a. Here, the switching curve for a transition between groups $g = 2$ and $g = 1$ is piecewise linear.



(a) Unconstrained system **(b)** Bounds $x_1^{\max} = 75$ and $x_1^{\max} = 92$.

Figure 6.5: Switching curves (black), for the system with parameters (6.12) and $c_1^+ = 2$, and transient trajectory (gray) for $x(0) = [3 \;\; 40 \;\; 80 \;\; 1]^\top$.

Adding maximal queue length constraints $x_1^{\max} = 75$ and $x_1^{\max} = 92$ to this model results in the switching curves depicted in Figure 6.5b. The figure also displays the optimal transient trajectory for $x(0) = \begin{bmatrix} 3 & 40 & 80 & 1 \end{bmatrix}^\top$. It can be seen that the switching curves, originating from the queue level constraints, are located $\lambda_n \sigma_{j,n}$ below $x_n^{\max}$, as the queue length increases during the setup. The initial queue contents, for starting in group 1, are limited to $x_1(0) \leq x_1^{\max} - \lambda_1 \sigma_{2,1}$ and $x_2(0) \leq x_2^{\max} - \lambda_2 \sigma$.

Note that for a system with service period constraints, i.e., with at least one of the constraints (6.6a)-(6.6b), switching curves may not exist in general, as the switching points are affected by the constraints and will depend on the initial state. This is illustrated in Figure 6.6. This figure presents, for the system with parameters (6.12) and with maximal service period $\tau_1^{\max} = 15$, the optimal transient trajectory starting at $x(0) = \begin{bmatrix} 3 & 100 & 10 & 1 \end{bmatrix}^\top$ by the solid line and the optimal transient trajectory starting at $x(0) = \begin{bmatrix} 3 & 130 & 5 & 1 \end{bmatrix}^\top$ by the dotted line. For both trajectories, the first service period of queue 1 equals $\tau_1^{\max} = 15$. Therefore, although both trajectories converge during the first cycle, the switching points are different. Also the subsequent switches depend on the initial state. Note that for the dotted trajectory, the server omits service of queue 2 in the first cycle.
For an optimal transient policy, the switching curves can be used to indicate the switching moments. From our experiments we find that for $c_n^+ \mu_n \geq c_j^+ \mu_j$, queue n is always emptied and the optimal policy for $c_n^+ \mu_n = c_j^+ \mu_j$ is, as expected, a clearing policy (unless prohibited by restrictions (6.6a)-(6.6d)).

For the system without service period and capacity constraints, the switching curves can also be derived analytically. We present this for a system with setup periods only and, without loss of generality, $c_1^+ \mu_1 \geq c_2^+ \mu_2$. Therefore, we assume that queue 1

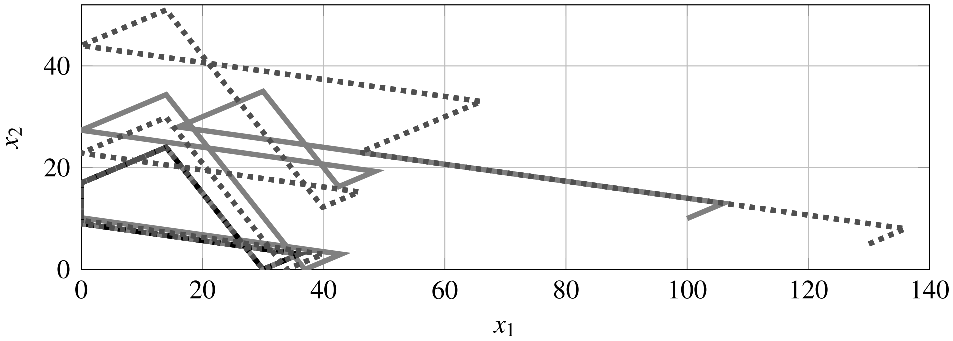
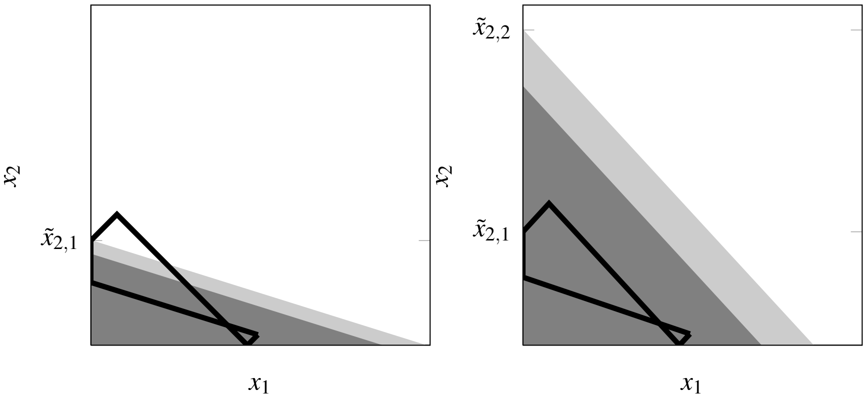


Figure 6.6: Optimal transient trajectory for $x(0) = [3 \;\; 100 \;\; 10 \;\; 1]^\top$, with $\tau_1^{\max} = 15$.

is always emptied once served. Also, the optimal steady-state trajectory includes a slow-mode (while serving queue 1), so that the transient trajectory can converge to it in finite time. Furthermore, we assume that once the transient trajectory converges to the steady-state trajectory, the system remains on this trajectory during normal service. To derive the transient trajectory, convergence to the steady-state trajectory is regarded step by step. Here, a step indicates a setup and service period. First, the final step is considered, i.e., the optimal trajectory to converge to the steady-state trajectory after a single step. Second, the optimal trajectory to reach the final step is considered. This approach is continued and results in the switching curves. Consider the optimal steady-state trajectory depicted in Figure 6.7a with $\tilde{x}_{2,1}$ the content of queue 2 at which the server switches to serve queue 1.



(a) Optimal steady-state trajectory with 1-step convergence areas.

(b) Optimal steady-state trajectory with 2-step convergence areas.

Figure 6.7: Optimal steady-state trajectory with different convergence areas.

We suppose that the transient trajectory converges to the steady-state trajectory at the start of setting up to serve queue 1, i.e., $x_1 = 0$ and $x_2 = \tilde{x}_{2,1}$, which follows from the optimal periodic behavior. The slow-mode in the steady-state trajectory is the only part of the trajectory at which capacity is lost during service. Therefore, it is the only part at which the transient trajectory can ‘catch up’. Note that the trajectory can converge during the slow-mode or for some specific initial points it can converge at other points on the steady-state trajectory. However, once on the steady-state trajectory, the system will remain on this trajectory and therefore we also assume convergence at $x_1 = 0$ and $x_2 = \tilde{x}_{2,1}$ for these particular cases.

The queue contents for which convergence during service of queue 1 is possible are depicted by the (dark and light) gray areas depicted in Figure 6.7a. The corresponding states are given by

$$x_2 \leq \tilde{x}_{2,1} - \frac{\lambda_2}{\mu_1 - \lambda_1} x_1, \qquad \text{if } x_0 = 0 \wedge g = 1. \tag{6.13a}$$

The dark gray area depicts the queue contents for which the system converges to the steady-state trajectory after the setup to queue 1, i.e., $x_0 = \sigma_{2,1}$. These states are given by

$$x_2 \leq \tilde{x}_{2,1} - \lambda_2 \sigma_1 \left[1 + \frac{\lambda_1}{\mu_1 - \lambda_1} \right] - \frac{\lambda_2}{\mu_1 - \lambda_1} x_1, \quad \text{if } x_0 = \sigma_{2,1} \wedge g = 1. \tag{6.13b}$$

We denote the areas (6.13a) and (6.13b) as 1-step convergence areas, since the transient trajectory can converge after a single service step (serving queue 1) to the steady-state trajectory. Given an initial state satisfying (6.13a), queue 1 must be emptied within $\frac{1}{\lambda_2}(\tilde{x}_{2,1} - x_2(0))$ time units to converge to the steady-state trajectory. It is optimal to serve the queue at the highest possible rate. The optimal service time durations for the system to converge to the steady-state trajectory at once, i.e., the duration of serving queue 1 at maximal process rate $\tau_{1,1}^{\mu *}$ and duration of serving at arrival rate $\tau_{1,1}^{\lambda *}$, are given by

$$\tau_{1,1}^{\mu *} = \frac{x_1(t)}{\mu_1 - \lambda_1}, \tag{6.14a}$$

$$\tau_{1,1}^{\lambda *} = \frac{1}{\lambda_2} \left[\tilde{x}_{2,1} - (x_2(t) + \lambda_1 \tau_1^{\mu}) \right], \tag{6.14b}$$

where t is the time at which the server starts serving queue 1. It is possible for the system to start in area (6.13a) and converge to the steady-state trajectory in the second cycle (second time serving queue 1), or even later. Below we show that this behavior is not optimal. Consider the system with initial state satisfying (6.13a), and we assume that the system converges to the steady-state behavior in the second cycle with service rates (6.14). Then, the first cycle results in additional decision variables $\tau_1^{\mu}, \tau_1^{\lambda}, \tau_2^{\mu}, \tau_2^{\lambda}$ before convergence to the steady-state trajectory. Also, after this first cycle, the state of the system must satisfy (6.13a). While serving a queue there are two options, clear the queue at maximal rate (which determines τ_n^{μ}) and use a slow-mode or switch when the queue is not yet empty (no slow-mode, i.e., $\tau_n^{\lambda} = 0$). This leaves us with the following four cases, for each of which the optimal service periods are derived. For ease of exposition, the calculations are omitted.

- **Clear queues 1 and 2**
  When both queues are cleared, the duration of the slow-modes have to be

determined. Minimal costs are reached when $\tau_1^\lambda = \tau_1^{\lambda^*}$ and $\tau_2^\lambda = 0$. Therefore, after the first service period of queue 1 the transient trajectory has converged to the steady-state trajectory and follows it during the next cycle.

- **Clear queue 1**
  When queue 1 is cleared and a slow-mode while serving queue 2 is not allowed, the variables are τ_1^λ and τ_2^μ, as τ_1^μ is determined by clearing queue 1 and $\tau_2^\lambda = 0$ since the queue is not cleared. Minimal costs are reached when the service duration τ_2^μ is maximal, i.e., clearing queue 2, and $\tau_1^\lambda = \tau_1^{\lambda^*}$ as shown above. Therefore, the resulting transient behavior is identical to the first case.

- **Clear queue 2**
  If a slow-mode while serving queue 1 is not allowed and queue 2 is cleared, the variables are τ_1^μ and τ_2^λ. Minimal costs are reached when queue 1 is cleared and $\tau_2^\lambda = 0$, and are higher than the solutions above, except for $x_2(0) \leq x_2^* - \frac{\lambda_2}{\mu_1 - \lambda_1} x_1(0)$ where the costs are identical. In this case, the system starts exactly on the outer edge of the light gray area depicted in Figure 6.7a and no slow-modes are required for convergence to the steady-state trajectory during service of queue 1.

- **Do not clear queues**
  When no slow-modes are allowed in the first cycle, the durations τ_1^μ and τ_2^μ are the optimization variables. Here, the optimal solution is identical to the previous case, i.e., empty both queues.

This illustrates that if the system is in (6.13a), it is optimal to converge to the steady-state trajectory using (6.14). Once the system reaches the steady-state trajectory, it remains on this trajectory. Note that this is not always optimal for the system with backlog, as presented in Section 6.3. Furthermore, denote by $R_i(x_1(t), x_2(t))$ the inventory costs during service in step i by $R_i^\sigma(x_1(t), x_2(t))$ the costs during the setup period at the start of step i. Hence, for the system with initial state in (6.13b), the optimal costs are given by

$$\begin{aligned} J_p^* =& R_1^\sigma(x_1(0), x_2(0)) + R_1(x_1(0) + \lambda_1 \sigma_{2,1}, x_2(0) + \lambda_2 \sigma_{2,1}) - \\ & - J_w^*(\sigma_{2,1} + \tau_{1,1}^{\mu *} + \tau_{1,1}^{\lambda *}), \end{aligned}$$

Next, the service time durations of queue 2 that lead the system to (6.13b) are investigated. Figure 6.7b presents the 2-step convergence areas, i.e., area for which the system can converge in 2 steps (successively serving queue 2 and queue 1) to the steady state behavior. These areas are given by

$$x_2 \leq \tilde{x}_{2,2} - \frac{\lambda_1}{\mu_2 - \lambda_2} x_1, \qquad \text{if } x_0 = 0 \wedge g = 2, \tag{6.15a}$$

$$x_2 \leq \tilde{x}_{2,2} - \mu_2 \sigma_2 - \frac{\lambda_1}{\mu_2 - \lambda_2} x_1, \qquad \text{if } x_0 = \sigma_{1,2} \wedge g = 2, \tag{6.15b}$$

$$\tilde{x}_{2,2} = \frac{\mu_1 - \lambda_1}{\lambda_2}\left(\tilde{x}_{2,1} - \lambda_2\sigma_1\left[1 + \frac{\lambda_1}{\mu_1 - \lambda_1}\right]\right).$$

The dark area in Figure 6.7b, given by (6.15b), represents the states for which the system can reach the area (6.13b), given initial setup times $\sigma_{1,2}$. From area (6.15a), represented by the light and dark gray areas in Figure 6.7b, the area (6.13b) can be reached while serving queue 2 without initial setup time.

The transient costs, starting from (6.15a) are given by

$$J_p = R_2(x_1(0), x_2(0)) + R_1^{\sigma} + R_1 - J_w^* \delta(\tau_{2,2}^{\mu} + \tau_{2,2}^{\lambda} + \sigma_{2,1} + \tau_{1,1}^{\mu*} + \tau_{1,1}^{\lambda*}), \tag{6.16}$$

where, for ease of reading, the parameters of R_1 and R_1^{σ} are omitted, i.e., the queue contents after step 2 and the setup period in step 1, respectively. For optimal costs (6.16), the optimal service durations during step 2 are required. Using a similar method as described above, we can conclude that slow-modes in the transient trajectory, except the final slow-mode to converge to the steady-state trajectory, result in non-optimal trajectories and are therefore not taken into account. Therefore, for the system in (6.15a), the service period in step 2 ($\tau_{2,2}^{\mu}$) is required that optimizes (6.16). This service period has a lower bound $\underline{\tau}_{2,2}^{\mu}$ to ensure reach area (6.13a) is reached, and an upper bound $\bar{\tau}_{2,2}^{\mu}$ which is the time required to empty queue 2, i.e.,

$$\underline{\tau}_{2,2}^{\mu} \leq \tau_{2,2}^{\mu} \leq \bar{\tau}_{2,2}^{\mu}, \tag{6.17}$$

$$\underline{\tau}_{2,2}^{\mu} = \frac{x_2(0) + \frac{\mu_2 - \lambda_2}{\lambda_1} x_1(0) - \tilde{x}_{2,1} + \lambda_2\sigma_{2,1} + \frac{\lambda_2}{\mu_1 - \lambda_1}\lambda_1\sigma_{2,1}}{\mu_2 - \lambda_2 - \frac{\lambda_1\lambda_2}{\mu_1 - \lambda_1}} - \frac{x_1(0)}{\lambda_1}, \tag{6.18}$$

$$\bar{\tau}_{2,2}^{\mu} = \frac{x_2(0)}{\mu_2 - \lambda_2}. \tag{6.19}$$

Then, the optimal service period $\tau_{2,2}^{\mu*}$ for starting in area (6.15a) is given by

$$\begin{aligned}\tau_{2,2}^{\mu*}(x_1(0), x_2(0)) = \min_{\tau_{2,2}^{\mu}} R_2(x_1(0), x_2(0)) + R_1^{\sigma} + R_1 - \\ - J_w^*(\tau_{2,2}^{\mu} + \sigma_{2,1} + \tau_{1,1}^{\mu*} + \tau_{1,1}^{\lambda*}),\end{aligned} \tag{6.20}$$

subject to (6.17). By continuing this step-by-step approach, the optimal service periods for all states can be derived and these can be presented as switching curves.

Alongside the switching curves, for an optimal transient policy, also the optimal initial group (given contents $x(0)$), if it is not predefined, can be derived. Together with the switching curves, this gives the policy for optimal transient behavior given initial queue contents. Once the optimal initial state is known, the queues are served until a switching point is reached, switch to the successive group, until converging to the optimal steady-state trajectory. If all initial states are allowed, the states with $x_0(0) > 0$ are of course not optimal. Therefore, a comparison of the transient costs for the system with $x_0(0) = 0$ and $g(0) = 1$ or $g(0) = 2$ results in the optimal initial state. For the system with parameters (6.12), the optimal initial states are presented in Figure 6.8, along with the switching curves. For initial queue contents in the gray area the optimal initial state is $x(0) = \begin{bmatrix} 0 & x_1 & x_2 & 1 \end{bmatrix}^\top$, and $x(0) = \begin{bmatrix} 0 & x_1 & x_2 & 2 \end{bmatrix}^\top$ otherwise.

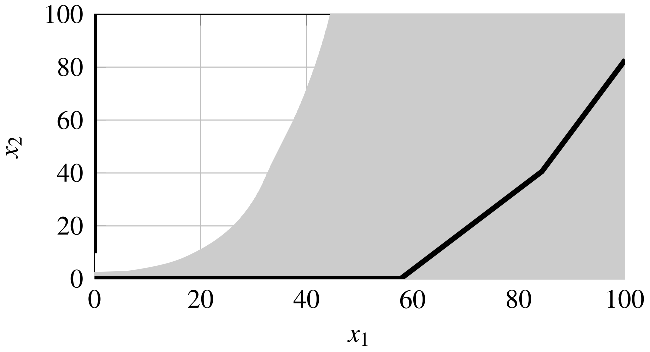


Figure 6.8: Switching curves (black) and optimal initial group for the system with parameters (6.12) and $c_1^+ = 2$, $x_0(0) = 0$, $g(0) = 1$ in the gray area and $g(0) = 2$ otherwise.

Also for the system with parameters (6.12), where $c_1^+ = 2$ and with queue length constraints $x_1^{\max} = 75$ and $x_1^{\max} = 92$, the optimal initial groups, with $x_0(0) = 0$, are presented in Figure 6.9. The dark gray area indicates that $g(0) = 1$ is optimal and the light gray area indicates optimal group $g(0) = 2$. For the remaining area, no trajectories exist as these would violate the queue constraints.

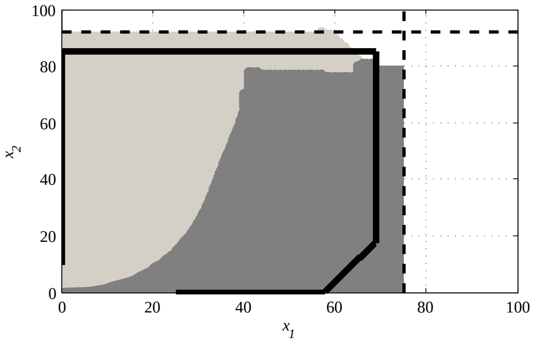


Figure 6.9: Switching curves (black) and optimal initial group for the system with parameters (6.12), $c_1^+ = 2$, $x_1^{\max} = 75$ and $x_1^{\max} = 92$. $x_0(0) = 0$, $g(0) = 1$ in the dark gray area and $g(0) = 2$ in the light gray area.

Note that, for the system without backlog, the optimal transient trajectory does not include idling of the server. Also, a slow-mode only occurs while on the steady-state trajectory or to converge to this trajectory. For the system with backlog, presented in Section 6.3, idling of the server and slow-modes can occur in the optimal transient behavior.

6.3 System with backlog

For the system with backlog, the transient costs are defined by

$$J_p = \liminf_{t\to\infty} \int_0^t [c_1^+ x_1^+(\tau) + c_1^- x_1^-(\tau) + c_2^+ x_2^+(\tau) + c_2^- x_2^-(\tau) + \\ + s_{2,1} v_1(\tau) + s_{1,2} v_2(\tau) - J_w^*] d\tau. \tag{6.21}$$

Deriving optimal transient trajectories for systems with backlog requires another approach as for the steady-state trajectory to derive the backlog and inventory levels per cycle, since Lemma 3.2.1 does not hold for each cycle, i.e., it is unknown if a queue content in a cycle becomes zero, and therefore the inventory and backlog can not be calculated by (3.10). Therefore, extra variables are introduced to calculate the inventory and backlog. Consider queue 1 that starts cycle c in group 1. The queue contents during this cycle are depicted in Figure 6.10.

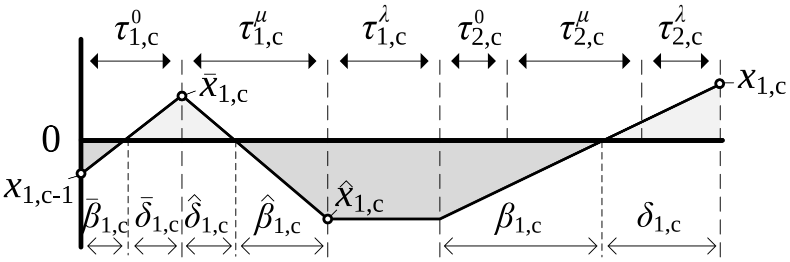


Figure 6.10: Evolution of x_1 during cycle c.

The periods in which the queue contents change, i.e., all periods except those serving at arrival rate, are divided into a part in which the content of queue n is positive, with a duration of $\delta_{n,c} \geq 0$, and a part in which the content is negative, with a duration of $\beta_{n,c} \geq 0$. For queue 1 it holds that

$$\bar{\delta}_{1,c} + \bar{\beta}_{1,c} = \tau_{1,c}^0, \tag{6.22a}$$
$$\hat{\delta}_{1,c} + \hat{\beta}_{1,c} = \tau_{1,c}^\mu, \tag{6.22b}$$
$$\delta_{1,c} + \beta_{1,c} = \tau_{2,c}^0 + \tau_{2,c}^\mu + \tau_{2,c}^\lambda. \tag{6.22c}$$

Furthermore, we denote by $\bar{x}_{1,c}$ the content of queue 1 in cycle c after the idle period $\tau_{1,c}^0$ and by $\hat{x}_{1,c}$ the content of queue 1 in cycle c after the period of maximal service $\tau_{1,c}^\mu$, see Figure 6.10. The queue levels at these time instances are divided into a positive and negative part, i.e.,

$$x_{n,c} = x_{n,c}^+ + x_{n,c}^-, \qquad n = 1,2, \tag{6.23a}$$
$$x_{n,c}^+ \leq \max(0, x_{n,c}), \qquad n = 1,2, \tag{6.23b}$$
$$x_{n,c}^- \leq \min(0, x_{n,c}), \qquad n = 1,2. \tag{6.23c}$$

Note that (6.23b) and (6.23c) are non-linear. One can see from Figure 6.10 that $\bar{\beta}_{1,c} = 0$ if $x_{1,c-1}^+ > 0$, as the queue content remains positive during the idle period.

Also, $x^+_{1,c-1} = 0$ if $\bar{\beta}_{1,c} > 0$, as the queue contents increase during the idle period. Therefore, and for similar reasons for all other periods, the following constraints hold

$$x^+_{1,c-1}\bar{\beta}_{1,c} = 0, \tag{6.24a}$$

$$\bar{x}^-_{1,c}\bar{\delta}_{1,c} = 0, \tag{6.24b}$$

$$\hat{x}^+_{1,c}\hat{\beta}_{1,c} = 0, \tag{6.24c}$$

$$\bar{x}^-_{1,c}\hat{\delta}_{1,c} = 0, \tag{6.24d}$$

$$\bar{x}^+_{1,c}\beta_{1,c} = 0, \tag{6.24e}$$

$$x^-_{1,c}\delta_{1,c} = 0. \tag{6.24f}$$

Note that these constraints are non-linear, and therefore can not be included in a QP problem. However, by adding these products of variables to the objective function, multiplied by a large gain to ensure that the constraints hold, minimization of the objective function results in (6.24).

Given these new variables, the total inventory of queue 1 in cycle c is defined by

$$\begin{aligned}
w^+_{1,c} &= \bar{w}^+_{1,c} + \hat{w}^+_{1,c} + \hat{x}^+_{1,c}\tau^\lambda_{1,c} + \tilde{w}^+_{1,c},\\
\bar{w}^+_{1,c} &= \tfrac{1}{2}\bar{\delta}_{1,c}(x^+_{1,c-1} + \bar{x}^+_{1,c}),\\
\hat{w}^+_{1,c} &= \tfrac{1}{2}\hat{\delta}_{1,c}(\bar{x}^+_{1,c} + \hat{x}^+_{1,c}),\\
\tilde{w}^+_{1,c} &= \tfrac{1}{2}\delta_{1,c}(\hat{x}^+_{1,c} + x^+_{1,c}),
\end{aligned}$$

and the total backlog of queue 1 in cycle c is defined by

$$\begin{aligned}
w^-_{1,c} &= \bar{w}^-_{1,c} + \hat{w}^-_{1,c} - \hat{x}^-_{1,c}\tau^\lambda_{1,c} + \tilde{w}^-_{1,c},\\
\bar{w}^-_{1,c} &= -\tfrac{1}{2}\bar{\beta}_{1,c}(x^-_{1,c-1} + \bar{x}^-_{1,c}),\\
\hat{w}^-_{1,c} &= -\tfrac{1}{2}\hat{\beta}_{1,c}(\bar{x}^-_{1,c} + \hat{x}^-_{1,c}),\\
\tilde{w}^-_{1,c} &= -\tfrac{1}{2}\beta_{1,c}(\hat{x}^-_{1,c} + x^-_{1,c}).
\end{aligned}$$

The evolution between the positive and negative queue contents is given by

$$\bar{x}^+_{1,c} = x^+_{1,c-1} + \lambda_1\bar{\delta}_{1,c}, \tag{6.25a}$$

$$\hat{x}^+_{1,c} = \bar{x}^+_{1,c} - (\mu_1 - \lambda_1)\hat{\delta}_{1,c}, \tag{6.25b}$$

$$x^+_{1,c} = \hat{x}^+_{1,c} + \lambda_1\delta_{1,c}, \tag{6.25c}$$

$$\bar{x}^-_{1,c} = x^-_{1,c-1} + \lambda_1\bar{\beta}_{1,c}, \tag{6.25d}$$

$$\hat{x}^-_{1,c} = \bar{x}^-_{1,c} - (\mu_1 - \lambda_1)\hat{\beta}_{1,c}, \tag{6.25e}$$

$$x^-_{1,c} = \hat{x}^-_{1,c} + \lambda_1\beta_{1,c}. \tag{6.25f}$$

For queue 2, we use a similar approach to derive the backlog and inventory during cycle c. The queue contents during this cycle are depicted in Figure 6.11. Denote

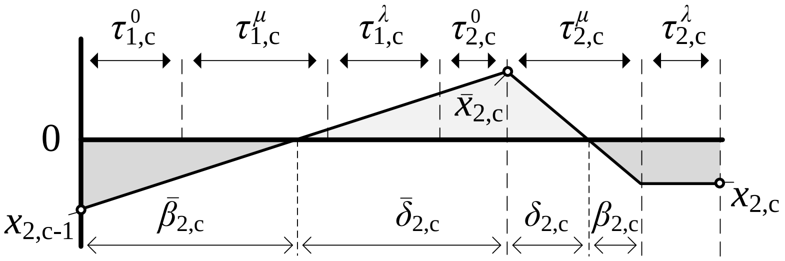


Figure 6.11: Evolution of x_2 during cycle c

by $\bar{x}_{2,c}$ the content of queue 2 in cycle c after the idle period $\tau^0_{2,c}$.
The periods during which queue 2 is not served at arrival rate are divided according to

$$\bar{\delta}_{2,c} + \bar{\beta}_{2,c} = \tau^0_{1,c} + \tau^\mu_{1,c} + \tau^\lambda_{1,c} + \tau^0_{2,c}, \tag{6.26a}$$
$$\delta_{2,c} + \beta_{2,c} = \tau^\mu_{2,c}. \tag{6.26b}$$

Then, the total inventory in cycle c is given by

$$\begin{aligned} w^+_{2,c} &= \bar{w}^+_{2,c} + w^+_{2,c} + \hat{x}^+_{2,c}\tau^\lambda_{2,c}, \\ \bar{w}^+_{2,c} &= \tfrac{1}{2}\bar{\delta}_{2,c}(x^+_{2,c-1} + \bar{x}^+_{2,c}), \\ \tilde{w}^+_{2,c} &= \tfrac{1}{2}\delta_{2,c}(\bar{x}^+_{2,c} + x^+_{2,c}), \end{aligned}$$

and total backlog by

$$\begin{aligned} w^-_{2,c} &= \bar{w}^-_{2,c} + w^-_{2,c} - \hat{x}^-_{2,c}\tau^\lambda_{2,c}, \\ \bar{w}^-_{2,c} &= -\tfrac{1}{2}\bar{\beta}_{2,c}(x^-_{2,c-1} + \bar{x}^-_{2,c}), \\ \tilde{w}^-_{2,c} &= -\tfrac{1}{2}\beta_{2,c}(\bar{x}^-_{2,c} + x^-_{2,c}). \end{aligned}$$

The evolution between the positive and negative queue contents is given by

$$\bar{x}^+_{2,c} = x^+_{2,c-1} + \lambda_2\bar{\delta}_{2,c}, \tag{6.27a}$$
$$x^+_{2,c} = \bar{x}^+_{2,c} - (\mu_2 - \lambda_2)\delta_{2,c}, \tag{6.27b}$$
$$\bar{x}^-_{2,c} = x^-_{2,c-1} + \lambda_2\bar{\beta}_{2,c}, \tag{6.27c}$$
$$x^-_{2,c} = \hat{x}^-_{2,c} - (\mu_2 - \lambda_2)\beta_{2,c}, \tag{6.27d}$$

and the constraints, derived similar to (6.24), are given by

$$x^+_{2,c-1}\bar{\beta}_{2,c} = 0, \tag{6.28a}$$
$$\bar{x}^-_{2,c}\bar{\delta}_{2,c} = 0, \tag{6.28b}$$
$$x^+_{2,c}\beta_{2,c} = 0, \tag{6.28c}$$
$$\bar{x}^-_{2,c}\delta_{2,c} = 0. \tag{6.28d}$$

Also these products of variables, multiplied by a large gain, are added to the objective function. Then, the transient costs (6.21) for the system with backlog, considering C cycles, can be written as

$$J_b(C) = C(s_{1,2} + s_{2,1}) + Q_b(C).$$

Here, $Q_b(C)$ is the solution to the quadratic programming problem, for C cycles, given by

$$Q_b(C) = \min_{\tau^{\mu}_{n,c}, \tau^{\lambda}_{n,c}} \sum_{n=1}^{2} \sum_{c=1}^{C} \left[c_n^+ w_{n,c}^+ + c_n^- w_{n,c}^- - J_w^*(\sigma_{n,j} + \tau^{\mu}_{n,c} + \tau^{\lambda}_{n,c}) \right] + \Lambda\Omega, \qquad n \neq j, \tag{6.29}$$

where Λ is a large gain an Ω is the sum of non-linear variables, i.e., the sum of the left hand sides of equations (6.24) and (6.28). Note that $\Lambda\Omega = 0$ ensures that the non-linear constraints are met. Objective function (6.29) is subject to constraints (6.22), (6.23a), (6.25), (6.26), and (6.27) and

$$x_{1,c} \leq x_1^{\max}, \tag{6.30a}$$

$$\bar{x}_{1,c} \leq x_1^{\max}, \tag{6.30b}$$

$$\bar{x}_{2,c} \leq x_2^{\max}, \tag{6.30c}$$

where (6.30) follows from (6.6d).

6.3.1 Illustrations

Consider the system with the following parameters

$$\begin{aligned} \lambda_1 &= 3, & \lambda_2 &= 1, \\ \mu_1 &= 8, & \mu_2 &= 9, \\ \sigma_{2,1} &= 1, & \sigma_{1,2} &= 3, \\ c_1^+ &= 2, & c_2^+ &= 1, \\ c_1^- &= 20, & c_2^- &= 10, \end{aligned} \tag{6.31}$$

and without setup costs or constraints on capacity or service periods. Figure 6.12 presents the optimal steady-state trajectory by the intersected line. The other trajectory, depicted with the solid line, starts and ends at the same state as the optimal steady-state trajectory and in between, each queue is served three times. Furthermore, this trajectory, which is also periodic, has lower costs than the optimal steady-state trajectory. As we consider the optimal steady-state trajectory derived in Section 3.2, periodic trajectories where each queue is served multiple times are not taken into account. Therefore, once the steady-state trajectory is reached, the system remains on this trajectory, and constraint (6.8) enforces the trajectory to converge to the steady-state trajectory.

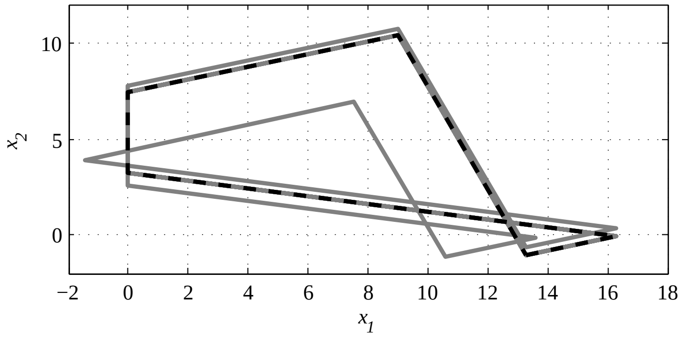


Figure 6.12: Steady-state trajectories where each flow is served once (intersected line) and each flow is served three times (solid line), for the system with parameters (6.31).

In Figure 6.13 the optimal steady-state trajectory (black intersected line) and three optimal transient trajectories are depicted. The solid (gray) line represents the transient trajectory with $x(0) = \begin{bmatrix} 3 & 10 & -10 & 1 \end{bmatrix}^\top$, where the server sets up to serve queue 2 directly after the setup to queue 1. The intersected line depicts the trajectory starting from $x(0) = \begin{bmatrix} 3 & -10 & -10 & 1 \end{bmatrix}^\top$. Here, after the setup, the server idles until $x_1 = 0$ and then converges to the steady-state cycle using a slow-mode. This example shows that optimal transient trajectories can include idling of the server, unlike the trajectories for the system without backlog. This is also intuitive, as idling of the server is the quickest way to remove backlog. A third trajectory, depicted by the dotted line in Figure 6.13, starts from $x(0) = \begin{bmatrix} 3 & -10 & 10 & 1 \end{bmatrix}^\top$ and also converges directly after emptying queue 1 and continuing service at arrival rate.

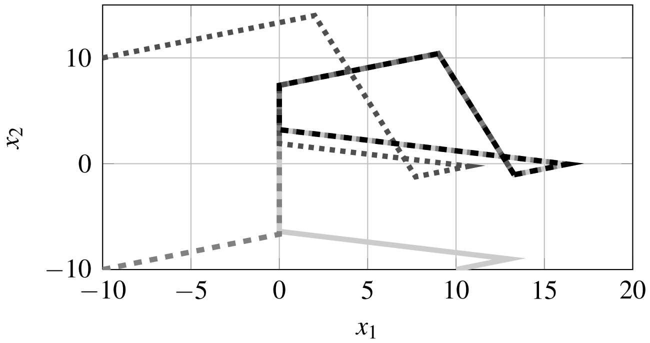


Figure 6.13: Optimal transient trajectories with different initial states, for the system with parameters (6.31).

In Figure 6.14, three optimal periodic trajectories are presented starting with a large backlog in queue 1. The initial state for the trajectory depicted by the gray solid line is $x(0) = \begin{bmatrix} 3 & -30 & -5 & 1 \end{bmatrix}^\top$, the trajectory depicted by intersected line $x(0) = \begin{bmatrix} 3 & -30 & 0 & 1 \end{bmatrix}^\top$ and the trajectory depicted by dotted line $x(0) = \begin{bmatrix} 3 & -30 & 15 & 1 \end{bmatrix}^\top$. It can be seen that the backlog is minimized as fast as possible, by switching to serve queue 2 and serving this queue or idling until $x_2 = 0$. Note that, for convergence to the steady-state trajectory, the server can switch earlier to serve queue 1. However, this reduction in time does not outweigh the extra backlog costs. Moreover, when the trajectory reaches the origin, the server can immediately switch to serve queue 1, resulting in a lower total inventory. For this system, this reduction in costs does not outweigh the cost reduction by elongating the cycle time. Therefore, queue 2 is served at arrival rate a little longer, i.e., until $x_2 = 2.06$.

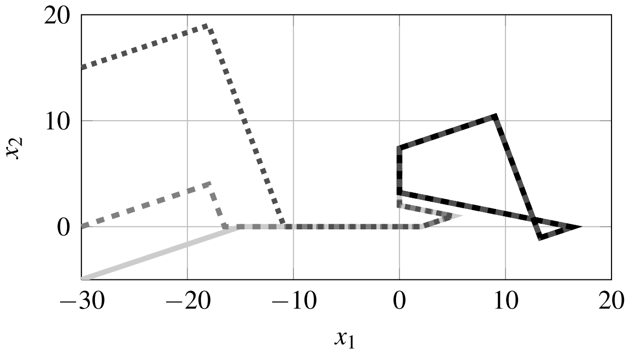


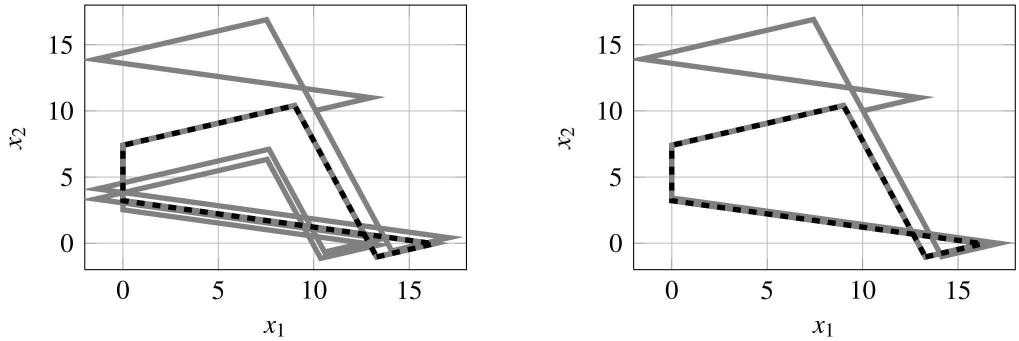
Figure 6.14: Optimal transient trajectories with different initial states, for the system with parameters (6.31).

In some cases, the optimal transient trajectory requires a few additional cycles to converge to the steady-state trajectory after the queue levels are directed to the vicinity of the steady-state trajectory. This trajectory has optimal costs, but another transient trajectory, which converges with less cycles to the steady-state trajectory and almost identical costs, might be desired by the operator, e.g., for simplicity of the trajectory. As an illustrative example, Figure 6.15 presents two different transient trajectories, both with initial state $\begin{bmatrix}3 & 10 & 0 & 1\end{bmatrix}^\top$. Figure 6.15a depicts the optimal transient trajectory, which requires four cycles to converge to the steady-state trajectory. In Figure 6.15b, a transient trajectory is depicted which converges after two cycles, with almost similar costs, i.e., a difference of 1,7%. This trajectory might be desired over the optimal trajectory, as the costs are almost similar and the trajectory converges faster to the steady-state trajectory.

Also, switching curves for the system with backlog, as presented for the system without backlog, provide no insight in the optimal transient behavior, due to this complex convergence for some of the optimal transient trajectories. To negate the occurrence of extra cycles just before converging, we add costs to the number of cycles required to converge. Denote by c_c the additional transient costs of requiring an extra cycle to converge to the steady-state behavior. Then, the transient costs $\bar{J}_b$ are denoted by

$$\bar{J}_b(C) = C(s_{1,2} + s_{2,1} + c_c) + Q_b(C). \tag{6.32}$$

Using (6.32), for the system with parameters (6.31) and $c_c = 5$, which is only 31%



(a) Convergence after 4 cycles, J_P^*=105.85.

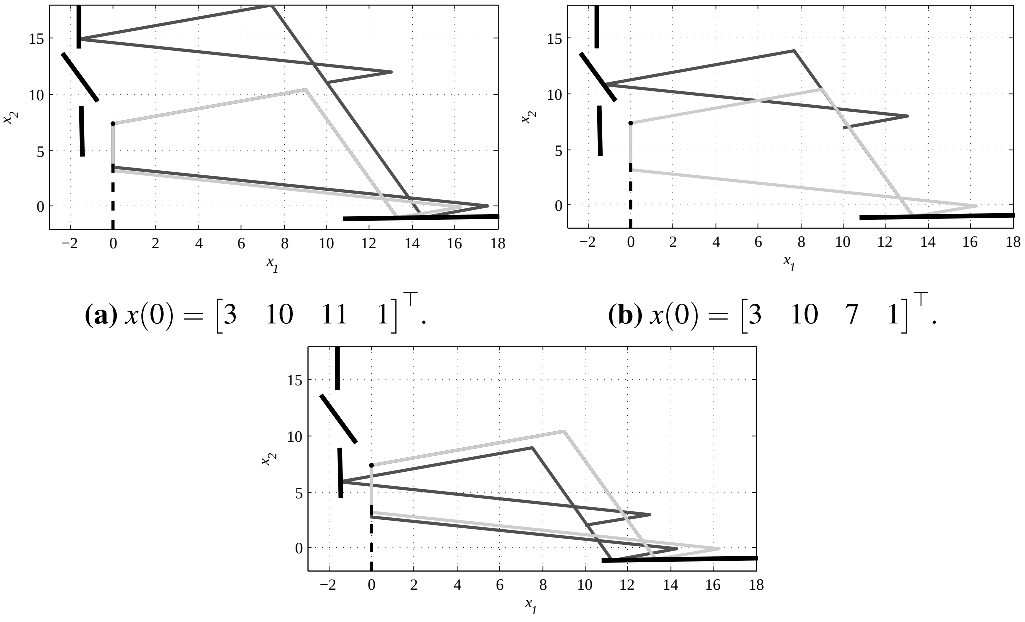
(b) Convergence after 2 cycles, J_P=107.64.

Figure 6.15: Two trajectories for $x(0) = [3\ \ 10\ \ 10\ \ 1]^\top$.

of the optimal steady-state costs J_w^*, the switching curves and three transient trajectories are depicted in Figure 6.16. These switching curves are depicted with the black solid lines, and are generated by optimizing trajectories for the system without backlog in the initial states. Also the optimal steady-state trajectory is depicted (gray) with corresponding switching points. It can be seen that the switching curves on the left side, which represent a transition between groups $g = 1$ and $g = 2$, consists of three line segments and single switching point on the steady-state trajectory. In Figure 6.16a a transient trajectory is depicted, which starts at $x(0) = \begin{bmatrix}3 & 10 & 11 & 1\end{bmatrix}^\top$ and converges to the steady-state trajectory during the slow-mode in the second cycle. The transient trajectory presented in Figure 6.16b, with initial state $x(0) = \begin{bmatrix}3 & 10 & 7 & 1\end{bmatrix}^\top$, converges to the steady-state trajectory during service of queue 2 in the first cycle. Due to allowed backlog, this is possible for a whole range of trajectories, as they switch at the second line segment of the switching curve. Finally, in Figure 6.16c a transient trajectory is depicted for the system with initial state $x(0) = \begin{bmatrix}3 & 10 & 0 & 1\end{bmatrix}^\top$. Note that the trajectory converges to the steady-state trajectory during service of queue 1 in the second cycle, while convergence was already possible during service of queue 1 in the first cycle. However, using two cycles results in lower costs (even with the added costs c_c). For a larger cost c_c, the use of the second cycle can be removed. Trajectories that reach a point on the intersected line, while serving queue 1, converge to the steady-state trajectory by using a slow-mode from that point.

6.4 Transient behavior of multi-queue switching servers

So far, the transient behavior of a two-queue switching server has been addressed. Extending this work to multi-queue servers imposes a couple of problems. First problem is the order of serving the queues until reaching the periodic behavior. If this order is not predefined, there exist lots of different transient service sequences, i.e., order of serving queues until reaching the periodic behavior. These sequences



(a) $x(0) = \begin{bmatrix}3 & 10 & 11 & 1\end{bmatrix}^\top$.

(b) $x(0) = \begin{bmatrix}3 & 10 & 7 & 1\end{bmatrix}^\top$.

(c) $x(0) = \begin{bmatrix}3 & 10 & 2 & 1\end{bmatrix}^\top$.

Figure 6.16: Switching curves, optimal steady-state trajectory and three transient trajectories.

can be generated similar to the sequence generation process of multi-queue switching servers, as presented in Section 4.4. Then for each sequence an optimization problem can be formulated and solved, similar to the method presented in Chapter 4. If the order of serving queues is predefined, for instance due to safety issues at traffic intersections or a fixed order of assembly at manufacturing systems, the optimization method follows directly from the method presented in this chapter.

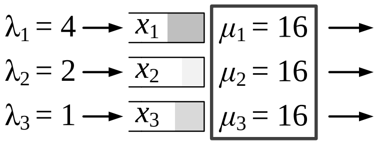
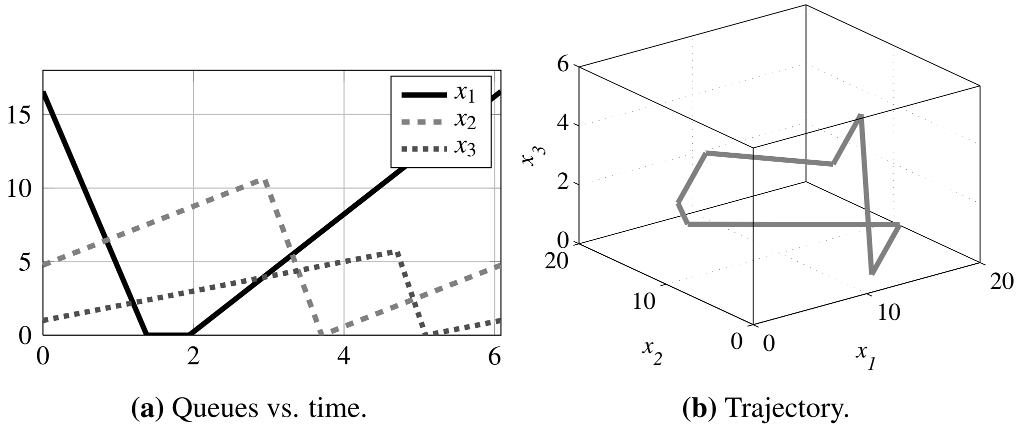


Figure 6.17: Three-queue switching server.

A second problem is the interpretation of the switching curves and optimal transient trajectories. If, for any initial state, the transient trajectories can be derived, a common policy (as depicted by the switching curves) is not easily found or does not exist. For example, let us consider the three-queue switching server presented in Section 4.6.1. The layout of this system is presented in Figure 6.17. The state of the 3-queue server is given by $x = \begin{bmatrix}x_0 & x_1 & x_2 & x_3 & g\end{bmatrix}^\top$. The arrival and service rates are indicated in the figure, all setup periods have a duration of 1 and the costs are given by $c_1 = 4$, $c_2 = 2$, $c_3 = 1$. Note that the costs differ from the system considered in Section 4.6.1, so that $c_n\mu_n$ is not constant. Furthermore, the server is restricted to serve only one queue at a time. Due to, for example, operator requirements, the service order of queues is fixed, i.e., assume without loss of generality that queues are served in the order 1, 2, 3 and then queue 1 again and so on. Also, backlog is not allowed, i.e., the queue lengths are nonnegative. Furthermore, no restrictions are imposed on the service periods or queue lengths. The optimal periodic behavior is presented in Figure 6.18.



(a) Queues vs. time. **(b)** Trajectory.

Figure 6.18: Optimal steady-state behavior of the 3-queue system.

Then, the optimal transient trajectory is derived similar as presented in Section 6.2, with the addition of service of queue 3. The combined switching points, derived by experiments, result in the *switching areas*. A switch to serve queue 2 or queue 3 occurs when queues 1 or 2 are emptied, i.e., $x = \begin{bmatrix}0 & 0 & \mathbb{R} & \mathbb{R} & 1\end{bmatrix}^\top$ or $x = \begin{bmatrix}0 & \mathbb{R} & 0 & \mathbb{R} & 2\end{bmatrix}^\top$, respectively. The switching area indicating a switch between serving queue 3 and setting up to serve queue 1 is presented in Figure 6.19. It can be seen that, queue 3 is, once served, not always emptied, as the switching area is not represented by $x = \begin{bmatrix}0 & \mathbb{R} & \mathbb{R} & 0 & 3\end{bmatrix}^\top$. This behavior is similar to the transient behavior of a two-queue system.

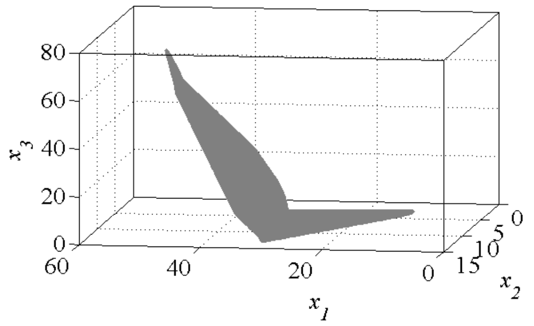


Figure 6.19: Switching area, switch from serving queue 3 to serve queue 1.

For this illustrative example, no constraints are imposed on the service or cycle times. Therefore, the server is allowed to switch to serve the next queue right after finishing a setup. This occurs for instance if $x(0) = \begin{bmatrix}1 & 200 & 200 & 200 & 1\end{bmatrix}^\top$. The corresponding optimal transient behavior is presented in Figure 6.20. It can be clearly seen that queue 3 is not served in the first cycle, i.e., after emptying queue 2, the system starts serving queue 1 again. To do this, first a setup to queue 3 is performed, followed by a setup to serve queue 1. This indicates that, if the order of service is not predefined, another sequence can result in better performance.

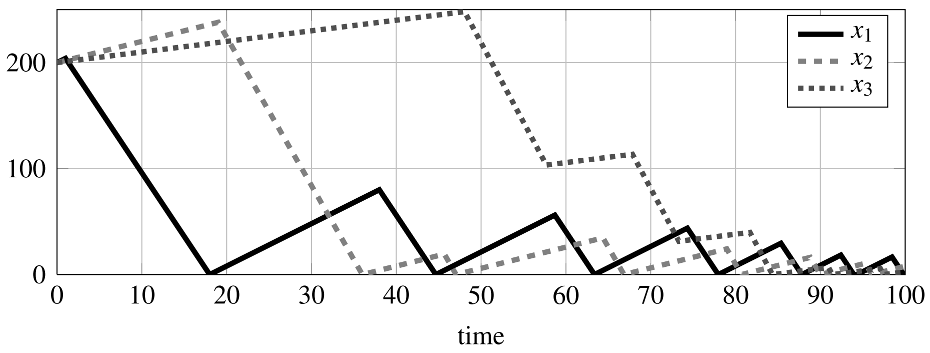


Figure 6.20: Optimal transient behavior for $x(0) = [1\ \ 200\ \ 200\ \ 200\ \ 1]^\top$.

6.5 Summary

In this chapter, the transient behavior of a two-queue switching server has been investigated intensively. For the system without backlog, the transient costs can be easily derived and given a number of cycles, a QP problem is presented. The solutions of the QP problem for different cycles are compared and the best one results yields the optimal transient behavior, given that the transient trajectory has converged to the optimal periodic trajectory. Given this method, the optimal transient trajectory can be derived given any initial state of the system. Next, for the unconstrained system, we combined the switching points, i.e., states at which the system in the optimal transient behavior switches to serve another queue. These combined points resulted into switching curves. Given the switching curves, the system knows exactly when to switch between queues to result in minimal transient costs. By bounding the queue lengths, the switching curves are also bounded. For a system with constraints on cycle time or service periods, switching curves may not exist in general, as the switching points are affected by the constraints and will depend on the initial state.

For a system with backlog, the QP problem formulation is more complex. Deriving the total amount of inventory and backlog is more tedious as for the periodic behavior (Chapter 3), as the queue content is not always zero once in a cycle. For a given initial state, the optimal transient behavior can be derived. The final state of the transient behavior is fixed, given the periodic behavior. This ensures that the transient trajectory converges to the periodic trajectory, as periods with multiple cycles, which is not allowed, can result in better performance. The resulting optimal trajectories showed complex switching behavior close to the periodic behavior which made the derivation of general switching curves impossible. However, the performance gain by the complex switching behavior is minimal compared to the switching considering less cycles. Therefore, additional costs on the number of cycles required to converge to the periodic behavior have been introduced. Using these additional costs, which just are a fraction of the average periodic costs, yield less complicated trajectories and also switching curves can be generated.

It is also shown that transient behavior for multi-queue switching servers can be derived similarly. However, for these servers, the order of serving queues is not fixed. Therefore, an excess of possible service orders exist to converge to the periodic behavior. Similar to the generation of all feasible sequences presented in Chapter 4, all possible sequences can be generated. Then, for each sequence an optimization problem can be formulated and solved and the best result yields the optimal transient behavior. Furthermore, as illustrated by the optimal periodic and transient behavior of a three-queue switching server, a common policy (as depicted by the switching curves for the two-queue system) is not easily found or does not exist.

Chapter 7

Observer design for a multi-queue single server

In Chapters 3-6, the control of switching servers has been studied. For a class of servers with two queues, the optimal periodic behavior has been derived in Chapter 3 and the optimal transient behavior in Chapter 6, for both unconstrained and constrained systems. Periodical behavior of multi-queue switching servers has been investigated in Chapter 4 and periodic behavior of networks of switching servers, without transportation times, has been studied in Chapter 5.

The controllers resulting from the aforementioned approaches are *central controllers* and determine the switching behavior of each server, based on the state of the entire system, i.e., global state information. Switching servers not only describe the behavior of man-made systems, but also describe the behavior of non-artificial systems, such as traffic intersection for example. For the latter systems, access to global state information is hardly ever the case in practice. This renders the *design of observers*, providing good estimates of the global state of the system, of crucial importance.

In this chapter the design of observers for a special class of piecewise affine hybrid systems (*PWAHS*), see [28], is investigated. This is motivated by multi-queue switching servers using a *clearing policy*, where one queue can be served at a time and the *order of serving the queues is fixed*. For this system, observers are investigated that, based on the measured input and output of the system, reconstruct the global state. An illustrative example of a manufacturing system that is used as running example throughout the chapter is presented in Figure 7.1 and we will refer to this system as the example system. This single switching server serves two queues $n = 1,2$. Fluid arrives at queue 1 with a constant arrival rate $\lambda_1 > 0$. After service in queue 1, the fluid moves to queue 2. The server can only serve one queue at a time and operates based on a clearing policy, i.e., it completely empties a queue before it switches to serve another queue. The service rate of queue n is denoted by $\mu_n > 0$. Note that, since the server switches when the queue is emptied, the

This chapter is partly based on [110].

queues are served at the maximal rate only, i.e., not also at arrival rate. Switching *to* queue n requires a setup time with duration $\sigma_n \geq 0$ and at least one setup time is non-zero, i.e., $\sigma_1 + \sigma_2 > 0$. Note that for ease of reading, the notation $\sigma_{i,j} = \sigma_j$ is used throughout the chapter, since the order of serving queues is fixed.

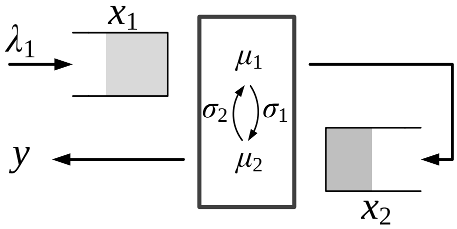


Figure 7.1: Two-queue switching server.

To model this example system, we use a continuous state that consists next to the queue contents x_n, $n = 1,2$, also of the remaining setup time at the server, denoted by x_0. Therefore, $x = \begin{bmatrix} x_0 & x_1 & x_2 \end{bmatrix}^\top \in \mathbb{R}_+^{N+1}$ with $N = 2$ being the number of queues and $\mathbb{R}_+ = [0, \infty)$. For each queue n the system has two modes, one for setting up to serve the queue and the other for serving the queue. Hence, this results in $M = 4$ modes (discrete states) in this case and the modes are denoted by $m \in \mathscr{M} := \{1, 2, \ldots, M\}$. The modes 1, 2, 3, and 4 represent setting up the server to serve queue 1, serving queue 1, setting up the server to serve queue 2, and serving queue 2, respectively. Note that the order in which the modes are traversed is fixed. The system evolves from mode 1 via modes 2, 3, and mode 4 back to mode 1 after which the cycle is repeated. Only at some times the output reveals information about the currently active mode. If measurable, the *output y* indicates the current mode of the system. For the example system, the only (measurement) information that is received is when the server is processing queue 2 in mode 4. Hence, the output is either 0 (system in modes 1, 2 or 3) or 4 if the system is in mode 4. Note that an output rate of μ_2 corresponds with $y = 4$.

The considered hybrid system is autonomous with the mode dynamics consisting of constant drift and the output within a mode being constant. For the example system, the modes 1 and 3 represent setup periods and in these modes the remaining setup time x_0 decreases with rate 1 and queue x_1 increases with rate λ_1. While serving queue 1 (mode 2), x_1 decreases with rate $\mu_1 - \lambda_1$ and x_2 increases with rate μ_2. In mode 4, i.e., serving queue 2, x_1 increases with rate λ_1 and x_2 decreases with rate x_2. In particular, this implies that all subsystems are unobservable, eliminating many currently available solutions for synthesizing hybrid observers proposed in the literature. Furthermore, the mode transitions are unknown a priori. However, the mode transitions are state-dependent, e.g., switch when the queue is empty or when the setup time has expired, and the order in which the modes are traversed is fixed and periodic, which are properties that will turn out to be useful. In addition, during a mode transition some specific state variables might exhibit jumps, e.g., the remaining setup times in the example system.

For this class of PWAHS, of which the full details are specified later, a methodology is proposed for designing continuous-time observers. This methodology consists of a few main steps. First, the system is sampled (with varying sampling periods) at so-called visible event times t_j^v, $j \in \mathbb{N}_{\geq 1}$, i.e., times at which the output changes during a mode transition. This results in a linear time-varying periodic system. Based on the resulting sampled system, a periodic discrete-time observer is derived with the guarantee that the observer's state converges asymptotically to the (original) system's state. Next, this observer is used as a stepping stone for designing an observer in continuous time. This requires the inclusion of additional modes in the observer structure and additional reset laws at visible event times to ensure the asymptotic recovery of the system's state. A formal proof of the asymptotic recovery of the system's state is provided. Via an example of a traffic application we demonstrate the effectiveness of the proposed observer.

A visual representation of the observer design process for the example system is presented in Figure 7.2. In the lower figure, the (observed) output is presented together with the visible event times. At these visible event times, the discrete-time observer estimates the current state of the system. This is illustrated in the upper figure for queue x_1 (the estimated state is denoted by $\hat{x}_1$). The crosses represent the estimated queue content at the visible event times. Although omitted in the figure, similar estimations are provided for the other states. Finally, a continuous-time observer is derived for which the estimated states at the visible event times coincide with the estimates of the discrete-time observer. The estimated state of the continuous-time observer is represented by the intersected line. This is basically a (smart) connection between the estimates of the discrete-time observer, based on the system dynamics. The details of each step can be found in the remainder of this chapter.

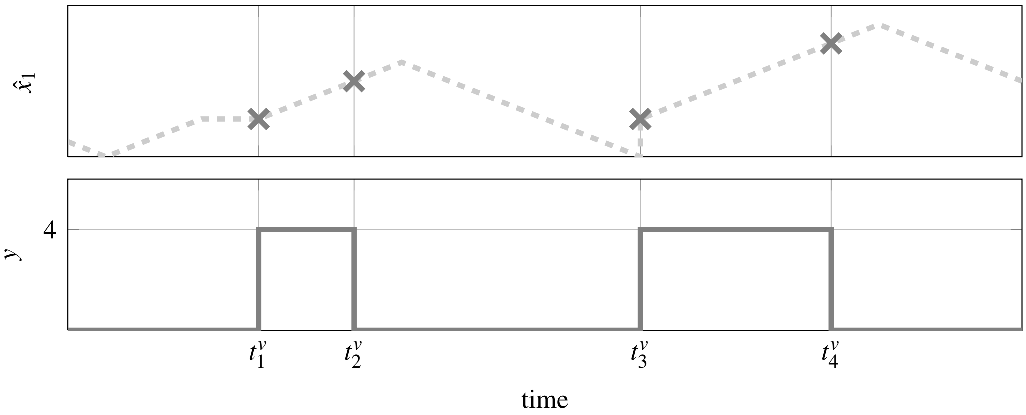


Figure 7.2: Visual representation of the observer design process: The output signal in the lower figure and the estimation of x_1 in the upper figure by the discrete-time observer (crosses) and continuous-time observer (dashed line).

The remainder of this chapter is organized as follows. Section 7.1 introduces the considered class of PWAHS and presents the dynamics of the example system. Section 7.2 presents the sampled system at the event times. In Section 7.3 the method for the observer design is presented. First, a discrete-time observer is presented for the sampled system. Next, the continuous-time observer is presented. In Section 7.4, both a signalized traffic intersection with three flows and a 4-queue switching server are presented for which an observer is derived. A summary is provided in Section 7.5.

Nomenclature
In this chapter we use $\{e_1, e_2, \dots, e_M\}$ as the standard orthonormal basis in $\mathbb{R}^M$ in which e_i is the vector which contains a 1 at the i-th entry, and zeros elsewhere. By $\mathbb{R}_+$ we denote the set of non-negative reals, i.e., $\mathbb{R}_+ := [0, \infty)$. Furthermore, the product of matrices is considered as a left multiplication, i.e., $\prod_{i=1}^{3} A_i = A_3 A_2 A_1$.

7.1 Class of piecewise affine hybrid systems

This section presents the dynamics of the class of piecewise affine hybrid systems (PWAHS) studied in this chapter. Before doing so, the remaining dynamics of the example system are presented.

7.1.1 Example continued

Recall that the example system is autonomous. In each mode $m \in \mathscr{M}$ the continuous state x has a constant drift vector, denoted by f_m, i.e., $\dot{x} = f_m$. For the example system, these drift vectors are given by

$$f_1 = f_3 = \begin{bmatrix} -1 \\ \lambda_1 \\ 0 \end{bmatrix}, \quad f_2 = \begin{bmatrix} 0 \\ \lambda_1 - \mu_1 \\ \mu_1 \end{bmatrix}, \quad f_4 = \begin{bmatrix} 0 \\ \lambda_1 \\ -\mu_2 \end{bmatrix}.$$

Furthermore, a transition occurs in modes 1 and 3 to the next modes 2 and 4, respectively, when $x_0 = 0$ indicating that the setup time has elapsed. As a result from the clearing policy, a transition from mode 2 to mode 3 occurs when $x_1 = 0$ and from mode 4 to mode 1 when $x_2 = 0$.

Due to the cyclic behavior in the way the modes are traversed, it holds at the k-th event time t_k, $k \in \mathbb{N}_{\geq 1}$, that the system switches from mode $m = ((k-1) \bmod M) + 1$ to the next mode $\delta(m)$ where $\delta : \mathscr{M} \to \mathscr{M}$ is given by

$$\delta(m) := 1 + (m \bmod M). \tag{7.1}$$

In addition, at the event time t_k the setup time x_0 instantaneously increases with constant $\omega_{m(t_k^-)} \in \mathbb{R}^+$, $m \in \mathscr{M}$, given by

$$\omega_1 = \omega_3 = 0, \quad \omega_2 = \sigma_2, \quad \omega_4 = \sigma_1,$$

i.e., we have a reset of the continuous state variable given by

$$x(t_k^+) = \begin{bmatrix} x_0(t_k^+) & x_1(t_k^+) & x_2(t_k^+) \end{bmatrix}^\top = \\ \begin{bmatrix} \omega_{m(t_k^-)} & x_1(t_k^-) & x_2(t_k^-) \end{bmatrix}^\top = x(t_k^-) + \omega_m e_1 \quad (7.2)$$

(as $x_0(t_k^-) = 0$, see also Lemma 7.1.7 below), where t_k^- denotes the time just before the k-th event. Note that e_1 is the unit vector with a 1 at the first entry and zeros elsewhere.

This reset law shows that discontinuities only appear in x_0, while x_n, $n = 1,2,...,N$, evolve continuously in time. Combining the above, leads to the following overall dynamics that can be compactly written as

$$\left.\begin{aligned} \dot{x} &= f_m \\ \dot{m} &= 0 \\ y &= h_m \end{aligned}\right\} \qquad \text{if } e_{\kappa_m}^\top x \geq 0, \qquad (7.3a)$$

$$\left.\begin{aligned} x^+ &= x + \omega_m e_1 \\ m^+ &= \delta(m) \end{aligned}\right\} \qquad \text{if } e_{\kappa_m}^\top x = 0 \qquad (7.3b)$$

with $\kappa_1 = \kappa_3 = 1$, $\kappa_2 = 2$ and $\kappa_4 = 3$ selecting the flow and jump sets (recall that e_{κ_m} is the κ_m-th unit vector in $\{e_1, e_2, \ldots, e_{N+1}\}$), i.e., $\kappa_1 = 1$ indicates that in mode 1 the first state variable (x_0) triggers the switch to another mode. Note that we expressed the example in terms of the modeling framework of jump-flow systems advocated in [43]. In fact, solutions/executions of the system under study can be interpreted in the sense of [43]. Initial conditions for this system are given by $x(0) \in \mathbb{R}_+^{N+1}$ with $x_0(0) = \omega_M$, and $m(0) = 1$. Besides we use the convention that $t_0 = 0$.

Measurement information regarding the knowledge of when the server is in mode 4, i.e., processing queue 2, is included via the output y, with

$$h_1 = h_2 = h_3 = 0, \qquad h_4 = 4.$$

Hence, as long as the system is in mode 4, this is directly seen in the output y. When the system is in one of the other modes $m \in \mathscr{M} \setminus \{4\}$, the output y is equal to 0 and no information is available from the server. Note that, actually, the service rate of queue 4 is measured in the example system. However, since the output $y = \mu_4$ indicates that the system is in mode 4, we use $y = 4$ in the remainder of this chapter.

7.1.2 General dynamics

In this section we provide the general description of the class of PWAHS under study, which includes the two-queue switching server discussed in the previous section as a particular case. Essentially, the general dynamics of the class of systems is given by (7.3) with continuous state

$$x = \begin{bmatrix} x_0 & x_1 & \ldots & x_N \end{bmatrix}^\top \in \mathbb{R}_+^{N+1}$$

with $N \in \mathbb{N}_{\geq 1}$, discrete state $m \in \mathscr{M} = \{1, 2, \ldots, M\}$ with $M \in \mathbb{N}_{\geq 1}$ and output $y \in \mathscr{M}_0 := \mathscr{M} \cup \{0\}$. The *data* of the system are given by the drift vectors $f_m \in \mathbb{R}^{N+1}$, the outputs $h_m \in \mathscr{M}_0$, and $\kappa_m \in \{1, 2, \ldots, N+1\}$ for each mode $m \in \mathscr{M}$, together with the reset parameters ω_m, $m \in \mathscr{M}$. In addition, we have for outputs h_m, $m \in \mathscr{M}$ that $h_m \in \{0, m\}$ for all $m \in \mathscr{M}$. Here, $h_m = m$ if the measured output indicates the current mode of the system, $h_m = 0$ otherwise. As in the two-queue switching server example, the general dynamics exhibits a *cycle*, i.e., a sequence of M consecutive modes being repeated over time. Some other special characteristics in the data being inherited from server-like systems in manufacturing and traffic applications are summarized below.

Assumption 7.1.1. *For all $m \in \mathscr{M}$ it holds that $e_{\kappa_m}^\top f_m < 0$ and $e_i^\top f_m \geq 0$ when $i \in \{1, \ldots, N+1\} \setminus \{\kappa_m\}$.*

This assumption guarantees that only one continuous state component decreases (being the one that also triggers the mode transition).

Assumption 7.1.2. *For all $m \in \mathscr{M}$*

$$e_1^\top f_m = -1, \qquad \text{if } \kappa_m = 1, \tag{7.4a}$$

$$e_1^\top f_m = 0, \qquad \text{if } \kappa_m \neq 1. \tag{7.4b}$$

This assumption expresses that x_0 is indeed a timer-related variable with only 0 and -1 as slopes. In case x_0 acts as a timer that triggers the next event (i.e., $\kappa_m = 1$) then $e_1^\top f_m = -1$, otherwise it is 0. The mode m for which $\kappa_m = 1$ corresponds to a setup for a switching server.

Assumption 7.1.3. *For all $m \in \mathscr{M}$ it holds that*

$$\omega_m > 0 \Leftrightarrow \kappa_{\delta(m)} = 1. \tag{7.5}$$

This assumption states that if a mode transition in mode m governs a jump in the state x_0, this state x_0 decreases in mode $\delta(m)$ and triggers the next mode transition (and vice versa).

Assumption 7.1.4.

$$\sum_{m=1}^{M} \omega_m > 0. \tag{7.6}$$

If translated in terms of a switching server, this assumption states that during a cycle of modes at least one setup of non-zero duration is present.

Assumption 7.1.5. *For all $m \in \mathscr{M}$ it holds that*

$$\kappa_m = 1 \Rightarrow \kappa_{\delta(m)} \neq 1. \tag{7.7}$$

This assumption expresses that a setup mode is followed by an operational mode $(\kappa_m \neq 1)$. Note that successive setup modes can be combined into a single setup mode.

Assumption 7.1.6. *There is at least one $m \in \mathcal{M}$ such that $h_m = m$.*

This assumption states that at least some information from the system is received.

Throughout the chapter we assume that all the mentioned assumptions are true (without further reference).

7.1.3 Basic results

In this section we derive some basic results for the class of PWAHS under study. To do so, let us denote, as before, by t_k the k-th time occurrence of a transition and $t_0 = 0$. Then we can prove the following lemmas, which are required for the observer design. At the end of a mode, for the example system, the remaining setup time is zero.

Lemma 7.1.7. *For any trajectory of the PWAHS (7.3) it holds that $x_0(t_k^-) = 0$ for all $k \in \mathbb{N}_{\geq 1}$.*

Proof. We prove this statement using induction. Suppose the statement holds for some $k \in \mathbb{N}_{\geq 1}$, i.e., $x_0(t_k^-) = 0$. At event time t_k, when going from mode m to mode $\delta(m)$, two situations can occur, being $\omega_m > 0$ or $\omega_m = 0$. In the first case, it holds that $x_0(t_k^+) = x_0(t_k^-) + \omega_m$. Then due to (7.3), (7.4a) and (7.5) it follows that $\dot{x}_0 = -1$ for $t \in [t_k, t_{k+1})$ until $x_0 = 0$. Hence $x_0(t_{k+1}^-) = 0$. In the second case, i.e., $\omega_m = 0$, no jump occurs and $x_0(t_{k+1}^-) = x_0(t_k^-)$ and $\dot{x} = 0$ for $t \in [t_k, t_{k+1})$ due to (7.4b) and (7.5). Therefore, $x_0(t_{k+1}^-) = x_0(t_k^-) = 0$. To complete the proof we need $x_0(t_1^-) = 0$. If $\omega_M > 0$, $x_0(t_1^-) = 0$ follows by the same reasoning as in the first case above. If $\omega_M = 0$, we immediately have $x_0(t_0) = 0$ and can use the reasoning in the second case to conclude $x_0(t_1^-) = 0$, thereby completing the proof. □

Lemma 7.1.8. *Consider $m \in \mathcal{M}$ such that $\omega_m > 0$. The dwell time in mode $\delta(m)$ is equal to ω_m.*

Proof. From Assumption 7.1.2 and Assumption 7.1.3 we know that if $\omega_m > 0$ we have $e_1^\top f_{\delta(m)} = -1$ and $\kappa_{\delta(m)} = 1$. Due to (7.3b) and Lemma 7.1.7 it holds that $x_0(t_k^+) = \omega_m$ for the event at time t_k where the system switches from mode m to mode $\delta(m)$. In addition, in mode $\delta(m)$ state x_0 decreases according to $\dot{x}_0 = -1$ until $x_0(t_{k+1}^-) = 0$. Hence, the dwell time in mode $\delta(m)$ is therefore given by

$$\frac{\omega_m}{-e_1^\top f_{\delta(m)}} = \omega_m.$$

□

For the class of PWAHS the following statements can be made regarding Zeno behavior and fixed points:

Lemma 7.1.9. *Zeno behavior, i.e., making an infinite number of jumps in a finite amount of time, is not present in system* (7.3).

Proof. Assumption 7.1.3 and Assumption 7.1.4 imply that there exists at least one mode in the cycle with $\omega_m > 0$, $m \in \mathscr{M}$. Lemma 7.1.8 shows that the dwell time in mode $\delta(m)$ is ω_m. Therefore, the dwell time of a cycle is bounded away from zero given the cyclic behavior, and no Zeno behavior occurs. □

Lemma 7.1.10. *System* (7.3) *does not contain any fixed points, i.e., the system always jumps to another mode in finite time.*

Proof. The condition $e_{\kappa_m}^\top f_m < 0$ in Assumption 7.1.1 guarantees that $e_{\kappa_m}^\top x$ decreases at constant rate, until it reaches the jump criterion $e_{\kappa_m}^\top x = 0$. Therefore, every mode is left in finite time. Since Zeno behavior is excluded, the system has no fixed points. □

Lemma 7.1.11. *System* (7.3) *is a positive system in the sense that if $x(0) \in \mathbb{R}_+^{N+1}$ with $x_0(0) = \omega_M$, and $m(0) = 1$, then $x(t) \in \mathbb{R}_+^{N+1}$ for all $t \in \mathbb{R}_+$.*

Proof. The proof follows similar reasoning as the proof of Lemma 7.1.7 using Assumptions 7.1.1 and 7.1.2. □

7.1.4 Visible and invisible modes and event times

In the remainder of this chapter we use the following notations in which we make a distinction between visible and invisible modes, and visible and invisible events.

A mode $m \in \mathscr{M}$ is called *visible* if $h_m = m$ and otherwise it is called *invisible*. The set of visible modes is denoted by $\mathscr{M}_v$, i.e.,

$$\mathscr{M}_v = \{m \in \mathscr{M} \mid h_m = m\}. \tag{7.8}$$

Transitions to and from visible modes are called *visible events* and transitions from invisible modes to invisible modes are called *invisible events*. To describe the visible events we define the set

$$\mathscr{V} = \mathscr{M}_v \cup \{m \in \mathscr{M} \mid \delta(m) \in \mathscr{M}_v\},$$

which is enumerated as $\{v_1, v_2, \ldots, v_V\} \subseteq \mathscr{M}$ with $1 \leq v_1 < v_2 < \ldots < v_V \leq M$. Using the above notation, if at time t_k the k-th event occurs jumping from mode $m(t_k^-)$ to mode $\delta(m(t_k^-))$, this event is visible if and only if $m(t_k^-) \in \mathscr{V}$. Hence, loosely speaking, we know when the system enters or leaves visible modes and we know when the corresponding events occur.

In the remainder we will use j as the visible event counter and denote visible event times by $t_j^v = t_{k(j)}$, $j \in \mathbb{N}_{\geq 1}$, where $k(j)$ translates the visible event j into the corresponding ordinary event k, i.e.,

$$k(j) = v_{1+(j-1 \text{ mod } V)} + \lfloor (j-1)/V \rfloor \cdot M, \tag{7.9}$$

where $\lfloor r \rfloor$ denotes the largest integer smaller than $r \in \mathbb{R}$. Due to the cyclic behavior in the way the (visible) nodes are traversed, it holds that visible event $v_{1+(j-1 \text{ mod } V)}$ has successive visible event $\delta_v(v_{1+(j-1 \text{ mod } V)})$ where $\delta_v : \mathscr{V} \to \mathscr{V}$, given by

$$\delta_v(v_l) := v_{1+(l \text{ mod } V)}, \qquad l = 1, 2, \ldots, V. \tag{7.10}$$

Example continued
For the example system, the set of visible modes is $\mathscr{M}_v = \{4\}$ and the set of visible events is $\mathscr{V} = \{3, 4\}$. The evolution of the modes of the example system and corresponding events and visible events is presented in Figure 7.3. Here, the visible modes are presented by the gray areas and it can be clearly seen that the events leading to or from visible modes are the visible events.

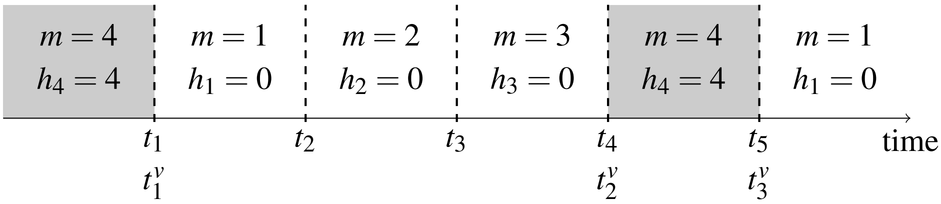


Figure 7.3: Mode evolution and corresponding events and visible events.

7.2 Sampling the hybrid system

One of the main ideas in the observer design methodology is to derive the desired continuous-time observer for system (7.3) from a discrete-time observer. To that end, we sample the system at the visible event times t_j^v, $j \in \mathbb{N}_{\geq 1}$. To easily derive the sampled data, we split its computation into three parts. First the system is sampled at all events t_k, $k \in \mathbb{N}_{\geq 1}$. Second, the state dimension is reduced by removing the timer variable x_0. In the third step the sampled system description is limited to only the visible events.

7.2.1 Sampling at all event times

Let $x(t_k^-)$ denote the state at time t_k just before the jump from mode $m(t_k^-)$ to mode $\delta(m(t_k^-))$. Sampling at all event times t_k, $k \in \mathbb{N}_{\geq 1}$, results in the system

$$x(t_{k+1}^-) = \tilde{A}_{m(t_k^-)} x(t_k^-) + \tilde{a}_{m(t_k^-)}, \tag{7.11a}$$

$$t_{k+1} = t_k + \tilde{C}_{m(t_k^-)} x(t_k^-) + \tilde{c}_{m(t_k^-)} \tag{7.11b}$$

in which

$$\tilde{A}_{m(t_k^-)} = I + \frac{f_{\delta(m(t_k^-))} e^\top_{\kappa_{\delta(m(k))}}}{-e^\top_{\kappa_{\delta(m(t_k^-))}} f_{\delta(m(t_k^-))}}, \qquad \tilde{a}_{m(t_k^-)} = \omega_{m(t_k^-)} A_{m(t_k^-)} e_1, \tag{7.12a}$$

$$\tilde{C}_{m(t_k^-)} = \frac{e^\top_{\kappa_{\delta(m(t_k^-))}}}{-e^\top_{\kappa_{\delta(m(t_k^-))}} f_{\delta(m(t_k^-))}}, \qquad \tilde{c}_{m(t_k^-)} = \omega_{m(t_k^-)} C_{m(t_k^-)} e_1 \tag{7.12b}$$

as can be derived from system (7.3). Note that the system (7.11) is periodic with period M in the sense that $\tilde{A}_{k+M} = \tilde{A}_k$ for $k \in \mathbb{N}_{\geq 1}$, and similar expressions hold for $\tilde{a}_k$, $\tilde{C}_k$ and $\tilde{c}_k$.

Remark 7.2.1. *Notice that from Assumptions 7.1.1 and 7.1.2 we obtain that $\tilde{A}_k$, $\tilde{a}_k$, $\tilde{C}_k$, and $\tilde{c}_k$ only contain non-negative elements, resulting in a positive system, which is to be expected given Lemma 7.1.11.*

The sampled system (7.11) can be used to write the system as a timed automaton [6] with a single clock, due to the linear dynamics and cyclic behavior. Then, the continuous state can be derived via a linear combination of the clock and the sampled state. However, observability and observer design for timed automata considers the discrete state reconstruction, see e.g., [77], which is straightforward for the class of systems under consideration. Furthermore, we are also interested in reconstructing the continuous state.

Example continued
By sampling the example system, we obtain a 4-periodic system for which the matrices in (7.12) are given as

$$\tilde{A}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{\mu_1}{\mu 1 - \lambda 1} & 1 \end{bmatrix}, \quad \tilde{a}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \tilde{C}_1 = \begin{bmatrix} 0 & \frac{1}{\mu_1 - \lambda_1} & 0 \end{bmatrix}, \quad \tilde{c}_1 = 0,$$

$$\tilde{A}_2 = \begin{bmatrix} 0 & 0 & 0 \\ \lambda_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \tilde{a}_2 = \begin{bmatrix} 0 \\ \sigma_2 \lambda_1 \\ 0 \end{bmatrix}, \quad \tilde{C}_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \quad \tilde{c}_2 = \sigma_2,$$

$$\tilde{A}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{\lambda_1}{\mu_2} \\ 0 & 0 & 0 \end{bmatrix}, \quad \tilde{a}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \tilde{C}_3 = \begin{bmatrix} 0 & 0 & \frac{1}{\mu_2} \end{bmatrix}, \quad \tilde{c}_3 = 0,$$

$$\tilde{A}_4 = \begin{bmatrix} 0 & 0 & 0 \\ \lambda_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \tilde{a}_4 = \begin{bmatrix} 0 \\ \sigma_1 \lambda_1 \\ 0 \end{bmatrix}, \quad \tilde{C}_4 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \quad \tilde{c}_4 = \sigma_1.$$

7.2.2 State reduction

From Lemma 7.1.7 we have $x_0(t_k^-) = 0$ for all $k \in \mathbb{N}_{\geq 1}$ and therefore this state is neglected. This results in the reduced state $\bar{x} = \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix}^\top = \begin{bmatrix} \mathbf{0}_N & I_N \end{bmatrix} x \in \mathbb{R}_+^N$, with $\mathbf{0}_N$ a zero column vector of length N and I_N the $N \times N$ identity matrix. For the reduced state, the dynamics at the event times become

$$\bar{x}(t_{k+1}^-) = \bar{A}_{m(t_k^-)}\bar{x}(t_k^-) + \bar{a}_{m(t_k^-)}, \tag{7.13a}$$

$$t_{k+1} = t_k + \bar{C}_{m(t_k^-)}\bar{x}(t_k^-) + \bar{c}_{m(t_k^-)}. \tag{7.13b}$$

where

$$\begin{aligned} \bar{A}_{m(t_k^-)} &= \begin{bmatrix} 0 & I \end{bmatrix} \tilde{A}_{m(t_k^-)} \begin{bmatrix} 0 & I \end{bmatrix}^\top, & \bar{a}_{m(t_k^-)} &= \begin{bmatrix} 0 & I \end{bmatrix} \tilde{a}_{m(t_k^-)}, \\ \bar{C}_{m(t_k^-)} &= \tilde{C}_{m(t_k^-)} \begin{bmatrix} 0 & I \end{bmatrix}^\top, & \bar{c}_{m(t_k^-)} &= \tilde{c}_{m(t_k^-)}. \end{aligned}$$

Note that due to this reduction no discontinuities in $\bar{x}(t)$ occur, as they only appear in x_0, see (7.2). Hence, $\bar{x}(t_k^-) = \bar{x}(t_k^+) = \bar{x}(t_k)$.

Example continued
For the example system, these reduced matrices are as follows:

$$\begin{aligned} \bar{A}_1 &= \begin{bmatrix} 0 & 0 \\ \frac{\mu_1}{\mu 1 - \lambda 1} & 1 \end{bmatrix}, & \bar{a}_1 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \bar{C}_1 &= \begin{bmatrix} \frac{1}{\mu_1 - \lambda_1} & 0 \end{bmatrix}, & \bar{c}_1 &= 0, \\ \bar{A}_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & \bar{a}_2 &= \begin{bmatrix} \sigma_2\lambda_1 \\ 0 \end{bmatrix}, & \bar{C}_2 &= \begin{bmatrix} 0 & 0 \end{bmatrix}, & \bar{c}_2 &= \sigma_2, \\ \bar{A}_3 &= \begin{bmatrix} 1 & \frac{\lambda_1}{\mu_2} \\ 0 & 0 \end{bmatrix}, & \bar{a}_3 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \bar{C}_3 &= \begin{bmatrix} 0 & \frac{1}{\mu_2} \end{bmatrix}, & \bar{c}_3 &= 0, \\ \bar{A}_4 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & \bar{a}_4 &= \begin{bmatrix} \sigma_1\lambda_1 \\ 0 \end{bmatrix}, & \bar{C}_4 &= \begin{bmatrix} 0 & 0 \end{bmatrix}, & \bar{c}_4 &= \sigma_1. \end{aligned}$$

7.2.3 Sampling at visible event times

Let $\bar{x}(t_j^v)$ denote the reduced state vector at t_j^v, the j^{th} visible event. From (7.13) it is clear that sampling at the visible events results in the system

$$\bar{x}(t_{j+1}^v) = A_{m(t_j^{v-})}\bar{x}(t_j^v) + a_{m(t_j^{v-})}, \tag{7.14a}$$

$$t_{j+1}^v = t_j^v + C_{m(t_j^{v-})}\bar{x}(t_j^v) + c_{m(t_j^{v-})}. \tag{7.14b}$$

Since we have V different visible events, the system (7.14) is a periodic linear system with period V. The system matrices in (7.14) for all $j \in \mathbb{N}_{\geq 1}$ are presented below. For $j \neq lV$, $l \in \mathbb{N}_{\geq 1}$, we have

$$A_j = \prod_{m=v_j}^{\delta_v(v_j)-1} \bar{A}_m, \tag{7.15a}$$

$$a_j = \sum_{m=v_j}^{\delta_v(v_j)-1} \left(\prod_{r=m+1}^{\delta_v(v_j)-1} \bar{A}_r \right) \bar{a}_m, \tag{7.15b}$$

$$C_j = \sum_{m=v_j}^{\delta_v(v_j)-1} \bar{C}_m \prod_{r=v_j}^{m-1} \bar{A}_r, \tag{7.15c}$$

$$c_j = \sum_{m=v_j}^{\delta_v(v_j)-1} \bar{c}_m + \sum_{m=v_j+1}^{\delta_v(v_j)-1} \bar{C}_m \sum_{r=v_j}^{m-1} \left(\prod_{s=r+1}^{m-1} \bar{A}_s \right) \bar{a}_r \tag{7.15d}$$

For $j = lV$, $l \in \mathbb{N}_{\geq 1}$, we have

$$A_j = \prod_{m=1}^{\delta_v(v_j)-1} \bar{A}_m \prod_{m=v_j}^{M} \bar{A}_m, \tag{7.15e}$$

$$a_j = \sum_{m=v_j}^{M+\delta_v(v_j)-1} \left(\prod_{r=m+1}^{M+\delta_v(v_j)-1} \bar{A}_r \right) \bar{a}_m, \tag{7.15f}$$

$$C_j = \sum_{m=v_j}^{M+\delta_v(v_j)-1} \bar{C}_m \prod_{r=v_j}^{m-1} \bar{A}_r, \tag{7.15g}$$

$$c_j = \sum_{m=v_j}^{M+\delta_v(v_j)-1} \bar{c}_m + \sum_{m=v_j+1}^{M+\delta_v(v_j)-1} \bar{C}_m \sum_{r=v_j}^{m-1} \left(\prod_{s=r+1}^{m-1} \bar{A}_s \right) \bar{a}_r \tag{7.15h}$$

with $A_{j+V} = A_j$, $j, \mathbb{N}_{\geq 1}$, and similar expressions for a_j, C_j, c_j and v_j.

Example continued
For the example system, we obtain a 2-periodic system (7.14) for which the matrices above are given by

$$A_1 = \begin{bmatrix} 1 & \frac{\lambda_1}{\mu_2} \\ 0 & 0 \end{bmatrix}, \qquad a_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 0 & \frac{1}{\mu_2} \end{bmatrix}, \qquad c_1 = 0,$$

$$A_2 = \begin{bmatrix} 0 & 0 \\ \frac{\mu_1}{\mu_1 - \lambda_1} & 1 \end{bmatrix}, \qquad a_2 = \begin{bmatrix} \sigma_2 \lambda_1 \\ \frac{\lambda_1 \mu_1 \sigma_1}{\mu_1 - \lambda_1} \end{bmatrix},$$

$$C_2 = \begin{bmatrix} \frac{1}{\mu_1 - \lambda_1} & 0 \end{bmatrix}, \qquad c_2 = \frac{\sigma_1 \mu_1}{\mu_1 - \lambda_1} + \sigma_2.$$

7.3 Observer design

Notice that information about the system's state is only received when visible events occur. Therefore, from the occurrence of visible events, the system's state is reconstructed.

To do so, an observer is build in two steps. First, starting from the dynamics (7.14), a linear time-varying (periodic) discrete-time observer is build to reconstruct the system's state at visible event times. Next, the dynamics (7.3) are used to make an open-loop prediction of the system's state, which is corrected (if necessary) at the next visible event time.

7.3.1 Discrete-time observer at visible event times

Our first goal is to build an observer which reconstructs the state at visible event times t_j^v, as described by the dynamics (7.14). To that end, we can use a Luenberger observer

$$\hat{\bar{x}}(t_{j+1}^v) = A_{m(t_j^{v-})}\hat{\bar{x}}(t_j^v) + a_{m(t_j^{v-})} + L_{m(t_j^{v-})}(t_{j+1}^v - \hat{t}_{j+1}^v), \tag{7.16a}$$

$$\hat{t}_{j+1}^v = t_j^v + C_{m(t_j^{v-})}\hat{\bar{x}}(t_j^v) + c_{m(t_j^{v-})}, \tag{7.16b}$$

where $L_1, L_2, \dots, L_V$ are the observer gains. Evolution of the observation error

$$\bar{e}(t_j^v) = \bar{x}(t_j^v) - \hat{\bar{x}}(t_j^v)$$

is given by

$$\bar{e}(t_{j+1}^v) = \left[A_{m(t_j^{v-})} - L_{m(t_j^{v-})}C_{m(t_j^{v-})}\right]\bar{e}(t_j^v).$$

Assumption 7.3.1. *There exist $L_1, L_2, \dots, L_V$ such that the observer* (7.16) *leads to*

$$\lim_{j\to\infty} \|\bar{e}(t_j^v)\| = 0.$$

Finding observer gains for this periodic system satisfying Assumption 7.3.1 is a known observer design problem, cf. [51, 75, 85]. For the illustrations presented in the next section, the periodicity of this system is exploited and a simple sequential algorithm presented in [50] is used to determine the time-varying observer gains to guarantee deadbeat convergence to zero at visible event times.

7.3.2 Continuous-time observer

Starting from the *reduced* state estimates at visible event times, as generated by the discrete-time observer (7.16), we will now provide a state estimate $\hat{x} \in \mathbb{R}^{N+1}$ of the full state x of (7.3), i.e., including x_0. In fact, we will guarantee that the estimated states $\hat{x}(t_j^{v+})$ just after visible events satisfy

$$\begin{bmatrix}\mathbf{0}_N & I_N\end{bmatrix}\hat{x}(t_j^{v+}) = \hat{\bar{x}}(t_j^v), \quad j \in \mathbb{N}_{\geq 1}, \tag{7.17}$$

indicating that just after the visible events the estimated states (without timer x_0) and the estimates $\hat{\bar{x}}(t_j^v)$ of the discrete-time observer (7.16) coincide. In addition, the dynamics (7.3) are used to make an open-loop prediction of the system's state after visible events. This open-loop prediction is updated using the observer (7.16) as soon as a new visible event happens.

However, notice that due to estimation errors, the predicted occurrence of the next visible event, $\hat{t}^v_{j+1}$ based on (7.16b), can be either sooner or later than the actual occurrence of the next visible event t^v_{j+1}. In the latter case ($t^v_{j+1} \leq \hat{t}^v_{j+1}$) the observer state can simply be updated according to (7.16), but in the former case (7.16) cannot (yet) be used, as the duration to the next visible event $t^v_{j+1} - \hat{t}^v_{j+1}$ is not known yet. Hence, in this case ($t^v_{j+1} > \hat{t}^v_{j+1}$), we have to determine the continuous-time observer dynamics for the period from predicted to actual occurrence of the next visible event. To do so, we introduce additional modes (called waiting modes) for the continuous-time observer. We therefore extend the set $\mathscr{M}$ of modes, by defining the set of observer modes as

$$\hat{\mathscr{M}} := \mathscr{M} \cup \left\{ v_l + \frac{1}{2} \middle| l = 1, 2, \ldots, V \right\},$$

where $v_l + \frac{1}{2}$, $l = 1, 2, \ldots, V$ are labels to denote the waiting modes. Furthermore, we define the mode transition map $\hat{\delta} : \hat{\mathscr{M}} \to \hat{\mathscr{M}}$, as

$$\hat{\delta}(\hat{m}) := \begin{cases} \hat{m} + \frac{1}{2} & \text{if } \hat{m} \in \mathscr{V} \\ \delta(\hat{m}) & \text{if } \hat{m} \in \mathscr{M} \setminus \mathscr{V} \\ \delta\left(\hat{m} - \frac{1}{2}\right) & \text{if } \hat{m} \in \hat{\mathscr{M}} \setminus \mathscr{M} \end{cases}$$

The waiting modes will only be used after an expected visible event which has not yet happened, i.e., when $\hat{t}^v_{j+1} \leq t < t^v_{j+1}$. For the additional waiting modes of the observer, we have to determine a drift vector for the state estimate. Notice from the observer (7.16) that at the visible event times the state estimate is updated according to (7.16a). That is, $L_{m(t^{v-}_j)}$ times the amount of time that the actual visible event is later than predicted is added. From this, a continuous evolution of the state estimate is derived which keeps $\hat{x}_0$ constant (at zero) and uses a drift vector of $L_{m(t^{v-}_j)}$ for the remaining state, i.e., $\dot{\hat{x}} = [0\ L^\top_{m(t^{v-}_j)}]^\top$, to avoid the need to reset the states at t^v_{j+1}, $j \in \mathbb{N}_{\geq 1}$ (although other choices guaranteeing (7.17) are fine as well). Furthermore, the output in the waiting mode is identical to the output of the preceding mode, i.e., $h_{\hat{m}} = h_{\hat{m}-\frac{1}{2}}$ for $\hat{m} \in \hat{\mathscr{M}} \setminus \mathscr{M}$.

Using the above reasoning, a continuous-time observer is deduced in the form of a jump-flow system [43]. The discrete-time observer (7.16) is embedded in the observer by including the state variables $\tilde{x} \in \mathbb{R}^N$ and $\tilde{t} \in \mathbb{R}$ in the continuous-time observer, which will satisfy $\tilde{x}(t) = \hat{\bar{x}}(t^v_j)$, $t \in [t^v_j, t^v_{j+1})$ (solutions to (7.16)), and $\tilde{t}(t) = t^v_j$, $t \in [t^v_j, t^v_{j+1})$. In between visible events $\tilde{x}$ and $\tilde{t}$ will be constant. This leads to the flow expressions for the continuous-time observer (7.18). When the measurement information equals the estimated output, i.e., $y = \hat{y}$ with $y = h_m$ and

$\hat{y} = h_{\hat{m}}$, see (7.3), the flow expressions are given by

$$\left.\begin{aligned} \dot{\hat{x}} &= f_{\hat{m}} \\ \dot{\hat{m}} &= 0 \\ \dot{\tilde{x}} &= 0 \\ \dot{\tilde{t}} &= 0 \\ \hat{y} &= h_{\hat{m}} \end{aligned}\right\} \quad \text{if } e_{\kappa_{\hat{m}}}^{\top}\hat{x} \geq 0 \wedge \hat{m} \in \mathscr{M} \wedge y = \hat{y}, \tag{7.18a}$$

$$\left.\begin{aligned} \dot{\hat{x}} &= \begin{bmatrix} 0 \\ L_{\delta_v^{-1}(\hat{m}-\frac{1}{2})} \end{bmatrix} \\ \dot{\hat{m}} &= 0 \\ \dot{\tilde{x}} &= 0 \\ \dot{\tilde{t}} &= 0 \\ \hat{y} &= h_{\hat{m}} \end{aligned}\right\} \quad \text{if } \hat{m} \in \hat{\mathscr{M}} \setminus \mathscr{M} \wedge y = \hat{y}. \tag{7.18b}$$

Note that (7.18a) describes the normal flow predictions based on model (7.3), while (7.18b) corresponds to the predictions in the waiting modes. The jump expressions for the normal flow predictions are given by

$$\left.\begin{aligned} \hat{x}^{+} &= \hat{x} + \omega_{\hat{m}} e_1 \\ \tilde{x}^{+} &= \tilde{x} \\ \hat{m}^{+} &= \hat{\delta}(\hat{m}) \\ \tilde{t}^{+} &= \tilde{t} \end{aligned}\right\} \quad \text{if } e_{\kappa_{\hat{m}}}^{\top}\hat{x} = 0 \wedge \hat{m} \in \mathscr{M} \wedge y = \hat{y}, \tag{7.18c}$$

following the normal jump expressions based on model (7.3). Furthermore, (7.18c) also describes that if a predicted visible event occurs before the actual visible event ($\hat{t}_{j+1}^{v} < t_{j+1}^{v}$), the discrete state jumps to a waiting mode.

If the measurement information differs from the estimated output, i.e., $y \neq \hat{y}$, a jump is required, since the system is in a different mode than the observer. Apart from an initial mode mismatch, the measurement difference $y \neq \hat{y}$ occurs via a change in y. In this case, a distinction is made between jumping from a mode without measurement information to a mode with measurement information, i.e., $y \neq \hat{y}$ and $y \neq 0$, and jumping from a mode with measurement information to a mode without measurement information, i.e., $y \neq \hat{y}$ and $y = 0$. In the former case, the measurement information provides information about the mode after the jump ($\hat{m}^{+} = y$). In the latter case, the mode after the jump is derived by using the cyclic mode transitions ($\hat{m}^{+} = \delta(\hat{y})$). The jump expressions are given by

$$\left.\begin{aligned} \zeta &= \tilde{t} + C_{p(y)}\tilde{x} + c_{p(y)} \\ \hat{x}^{+} &= \begin{bmatrix} 0 \\ A_{p(y)} \end{bmatrix} \tilde{x} + \begin{bmatrix} \omega_{\delta^{-1}(y)} \\ a_{p(y)} \end{bmatrix} + \begin{bmatrix} 0 \\ L_{p(y)} \end{bmatrix} (t - \zeta) \\ \hat{m}^{+} &= y \\ \tilde{x}^{+} &= \hat{\bar{x}}^{+} \\ \tilde{t}^{+} &= t \end{aligned}\right\} \text{if } y \neq \hat{y} \wedge y \neq 0, \tag{7.18d}$$

$$\left.\begin{aligned}\zeta &= \tilde{t} + C_{\delta^{-1}(\hat{y})}\tilde{x} + c_{\delta^{-1}(\hat{y})}\\ \hat{x}^+ &= \begin{bmatrix}0\\ A_{\delta^{-1}(\hat{y})}\end{bmatrix}\tilde{x} + \begin{bmatrix}\omega_{\delta(\hat{y})}\\ a_{\delta^{-1}(\hat{y})}\end{bmatrix} + \begin{bmatrix}0\\ L_{\delta^{-1}(\hat{y})}\end{bmatrix}(t-\zeta)\\ \hat{m}^+ &= \delta(\hat{y})\\ \tilde{x}^+ &= \hat{\bar{x}}^+\\ \tilde{t}^+ &= t\end{aligned}\right\} \text{if } y \neq \hat{y} \wedge y = 0, \tag{7.18e}$$

where we denoted $p(y) = \delta_v^{-1}(\delta^{-1}(y))$ and introduced ζ for ease of exposition. Recall that $\hat{\bar{x}}(t) = [0\ I]\hat{x}(t)$. Note that (7.18d) and (7.18e) describe the jump expressions at occurrence of a visible event. Based on some knowledge of initial conditions of the original system (7.3), we initialize the observer as $\hat{x}(0) \in \mathbb{R}_+^{N+1}$ with $\tilde{x}(0) = \hat{\bar{x}}(0)$ and $\tilde{t}(0) = 0$.

Proposition 7.3.2. *Suppose Assumption 7.3.1 holds. Then the continuous-time observer* (7.18) *asymptotically reconstructs the states $\bar{x}$ of the system* (7.3)*, i.e.,*

$$\lim_{t\to\infty} \|\bar{x}(t) - \hat{\bar{x}}(t)\| = 0.$$

Proof. Since the continuous-time observer was designed to satisfy $\|[0\ I]x(t_j^{v+}) - [0\ I]\hat{x}(t_j^{v+})\| = \|\bar{x}(t_j^v) - \hat{\bar{x}}(t_j^v)\|$, $j \in \mathbb{N}_{\geq 1}$, it suffices to show that there exists a constant P such that $\|\bar{x}(t) - \hat{\bar{x}}(t)\| \leq P\|\bar{x}(t_j^v) - \hat{\bar{x}}(t_j^v)\|$ for $t \in [t_j^v, t_{j+1}^v)$, $j \in \mathbb{N}_{\geq 1}$. The latter will follow from the observation that the observation error only changes when the observer is in a different mode than the actual system, as shown below.

Notice that (7.11) describes the evolution of the mode changes of the system, i.e., the state updates at the switching times t_k. Let $\hat{t}_k$ denote the switching times as predicted by the continuous-time observer. Then we have from (7.11b):

$$t_{k+1} - \hat{t}_{k+1} = t_k - \hat{t}_k + \tilde{C}_{m(t_k^-)}[x(t_k^-) - \hat{x}(t_k^-)]. \tag{7.19}$$

Since at time t_j^v we have $\hat{t}_{k(j)} = t_{k(j)}$, it follows from (7.19) that the duration of the mode difference is a linear function of the initial observer error, i.e., and so is the sum of the durations of mode differences. Finally, since during mode differences the rate of increase is bounded, and the number of invisible modes between two consecutive visible modes is finite, the observation error $\|\bar{x}(t) - \hat{\bar{x}}(t)\|$ on the interval $t \in [t_j, t_{j+1})$ can be upperbounded by $\|\bar{x}(t) - \hat{\bar{x}}(t)\| \leq P\|\bar{x}(t_j^v) - \hat{\bar{x}}(t_j^v)\|$.

Notice that from (7.19) we also have that $\lim_{k\to\infty} |t_k - \hat{t}_k| = 0$. □

Remark 7.3.3. *Using a deadbeat (or exact) observer, the complete state can be asymptotically reconstructed, i.e.,* $\lim_{t\to\infty} \|x(t) - \hat{x}(t)\| = 0$*. For non-deadbeat observers, peaking occurs in* $\|x_0(t) - \hat{x}_0(t)\|$ *due to mismatch in event times combined with system jumps* (7.2)*. However, in most manufacturing and traffic applications one is interested in the queue sizes or queue lengths, i.e.,* $\hat{\bar{x}}(t)$*.*

Remark 7.3.4. *As we observed in Lemma 7.1.11 and Remark 7.2.1, we are essentially dealing with a positive system. However, the observer (7.16) does not necessarily guarantee positivity of the state estimates $\hat{x}(t)$, $t \in \mathbb{R}$. Though the observer (7.18) is well defined and asymptotically recovers the state of the original system, from a physical point of view it would be better to have non-negative state estimates. As we show in the next section by means of an example, it is possible to derive observers that respect the positivity property by generating non-negative state estimates. Note that observer gains satisfying $(A_v - L_vC_v) \geq 0$ and $L_vC_v \geq 0$, $v \in \mathscr{V}$, suffice since*

$$\hat{\bar{x}}(t_{j+1}^{v-}) = (A_{m(t_j^{v-})} - L_{m(t_j^{v-})}C_{m(t_j^{v-})})\bar{x}(t_j^{v-}) + L_{m(t_j^{v-})}C_{m(t_j^{v-})}x(t_j^{v-}) + a_{m(t_j^{v-})}.$$

In fact, designing directly positive observers is one of the questions for future research.

One direction to pursue in this context is considering the dynamics (7.14) only once every J time-instances and lift the system (see, for instance, [27]) leading to a linear time-invariant positive system. The positive observation problem for linear discrete time-invariant positive systems has been dealt with in [83], where a necessary and sufficient condition for the existence and the design of a positive linear observer of Luenberger form has been given by means of the feasibility of a linear program (LP). This could form an interesting starting point to obtain positive discrete-time observers of the form (7.16a) (which is doable under certain assumptions). The step towards a continuous-time observers could follow then mutatis mutandis *the line of reasoning as indicated above.*

An alternative solution leading to positive estimates is to use a projection of $\hat{\bar{x}}(t)$ on the positive cone $\Omega = \mathbb{R}_+^N$ as the estimated state instead of $\hat{\bar{x}}(t)$, i.e., use $P_\Omega\hat{\bar{x}}(t)$ with $P_\Omega : \mathbb{R}^N \to \mathbb{R}_+^N$, given for $z \in \mathbb{R}^N$

$$(P_\Omega z)_i = \max(0, z_i).$$

This projected estimate also asymptotically recovers the true state, i.e.,

$$\lim_{t\to\infty} \|P_\Omega\hat{\bar{x}}(t) - \bar{x}(t)\| = 0.$$

7.4 Illustrations

To demonstrate the observer design, two illustrations are presented below. First, the observer design of a signalized 3-queue traffic intersection is illustrated. Second, observer design of a 4-queue re-entrant switching server is presented.

7.4.1 Traffic intersection

Consider a signalized T-junction consisting of three flows of cars, see Figure 7.4. Each flow can go into two directions. For this example we assume that each flow has a single signal, i.e., if a car receives a green light it can move in two different directions. Therefore, it is not possible to give multiple flows a green light simultaneously. The flows are served in order 1, 2, 3, and then back to flow 1 after which the cycle is repeated. The intersection uses a clearing policy, i.e., it completely empties the queue of a flow before switching to serve the next flow. Switching to serve cars from flow i requires a clearing/setup time σ_i to make sure all vehicles from the previously served flow have cleared the intersection. Vehicles arrive at flow i with arrival rate λ_i and are served with process rates μ_i, $i = 1,2,3$. A sensor measures the crossing of vehicles in lane 1.

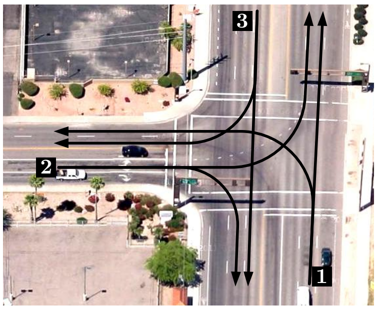


Figure 7.4: Signalized T-junction containing three vehicle flows 1-3.

The dynamics can be written in the form (7.3) with

$$f_1 = f_3 = f_5 = \begin{bmatrix} -1 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix},$$

$$f_2 = \begin{bmatrix} 0 \\ \lambda_1 - \mu_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}, \quad f_4 = \begin{bmatrix} 0 \\ \lambda_1 \\ \lambda_2 - \mu_2 \\ \lambda_3 \end{bmatrix}, \quad f_6 = \begin{bmatrix} 0 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 - \mu_3 \end{bmatrix}$$

$$\omega_1 = \omega_3 = \omega_5 = 0, \quad \omega_2 = \sigma_2, \quad \omega_4 = \sigma_3, \quad \omega_6 = \sigma_1,$$
$$\kappa_1 = \kappa_3 = \kappa_5 = 1, \quad \kappa_2 = 2, \quad \kappa_4 = 3, \quad \kappa_6 = 4,$$
$$h_1 = h_3 = h_4 = h_5 = h_6 = 0, \quad h_2 = 2.$$

The modes 1,3 and 5 denote setting up to serve flow 1,2 and 3, respectively. Modes 2,4 and 6 denote serving flow 1,2 and 3, respectively. Writing this system in the form (7.14) gives

$$A_1 = \begin{bmatrix} 0 & 0 & 0 \\ \frac{\lambda_2}{\mu_1 - \lambda_1} & 1 & 0 \\ \frac{\lambda_3}{\mu_1 - \lambda_1} & 0 & 1 \end{bmatrix}, \qquad C_1 = \begin{bmatrix} \frac{1}{\mu_1 - \lambda_1} & 0 & 0 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 1 & \frac{\lambda_1\mu_3}{(\mu_2-\lambda_2)(\mu_3-\lambda_3)} & \frac{\lambda_1}{\mu_3-\lambda_3} \\ 0 & \frac{\lambda_2\lambda_3}{(\mu_2-\lambda_2)(\mu_3-\lambda_3)} & \frac{\lambda_2}{\mu_3-\lambda_3} \\ 0 & 0 & 0 \end{bmatrix}, \qquad C_2 = \begin{bmatrix} 0 & \frac{\mu_3}{(\mu_2-\lambda_2)(\mu_3-\lambda_3)} & \frac{1}{\mu_3-\lambda_3} \end{bmatrix}.$$

Note that a_i and c_i are omitted as they cancel out in the observer error dynamics. Also note that though the pairs (A_1,C_1) and (A_2,C_2) are unobservable, we can build a periodic deadbeat observer by using the observer gains

$$L_1 = \begin{bmatrix} 0 & \lambda_2 & \lambda_3 \end{bmatrix}^\top, \tag{7.20a}$$

$$L_2 = \begin{bmatrix} \lambda_1 & \frac{\lambda_2\lambda_3}{\mu_3} & 0 \end{bmatrix}^\top. \tag{7.20b}$$

Using these gains, the matrices $A_v - L_vC_v$ and L_vC_v $(v = 1,2)$ are positive matrices, yielding a positive observer. In this example the observer estimation starts at $t = 50$. For parameters

$$\lambda = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^\top, \qquad \mu = \begin{bmatrix} 8 & 10 & 12 \end{bmatrix}^\top,$$
$$\sigma = \begin{bmatrix} 5 & 10 & 15 \end{bmatrix}^\top,$$

and initial estimated state $\hat{x}(50) = \begin{bmatrix} 0 & 70 & 20 & 30 \end{bmatrix}^\top$ and mode $\hat{m}(50) = 4$, i.e., serving vehicles from flow 2, a simulation result is presented in Figure 7.5. The queue lengths at the intersection are presented by dashed lines and the estimated queue lengths are presented by solid lines. Since the measurement information equals the estimated output, i.e., $y(50) = \hat{y}(50)$, the observer dynamics are given by (7.18a)–(7.18e) until measurement and estimated output differ. This occurs at $t = 70.7$, when the system starts serving vehicles from queue 1. At this time instant $y \neq \hat{y}$ and the system jumps according to (7.18d). According to the initial estimated state $\hat{\bar{x}}(50)$, this event was predicted at $\tilde{t} = 81.7$ with $\tilde{x}(\tilde{t}) = \begin{bmatrix} 101.7 & 58.3 & 15 \end{bmatrix}^\top$. Therefore, $\hat{\bar{x}}(70.7^+) = \begin{bmatrix} 90.7 & 52.8 & 15 \end{bmatrix}^\top$, causing jumps in $\hat{x}_2$ and $\hat{x}_3$. Also, the estimated mode changes to serving products from flow 1.

The second visible event occurs at $t = 79.2$ when $x_1 = 0$, also earlier than predicted since $\hat{x}_1$ is estimated too large. At this moment $y \neq \hat{y}$ and $y = 0$, therefore jump (7.18e) results in $\hat{x}_1(79.2^+) = 0$. Note that at this moment both $\hat{x}_1(t)$ and $\hat{x}_3(t)$ are correctly estimated. The observer predicts the next visible event at $\tilde{t} = 137$. However, since $\hat{x}_2(79.2^+) < x_2(79.2^+)$ the actual event occurs later. Therefore the observer switches to the additional waiting mode (7.18b) via jump (7.18c). At $t = 140.3$, detection of the next visible event, the estimated queue levels have converged exactly to the actual queue levels, due to the deadbeat observer.

7.4.2 Switching server

Another example presents the observer design and performance of a switching server with four queues. A graphical representation of the server is presented in Figure 7.6. Fluid served in queue 1 requires another service at the server in queue 2, i.e., this is

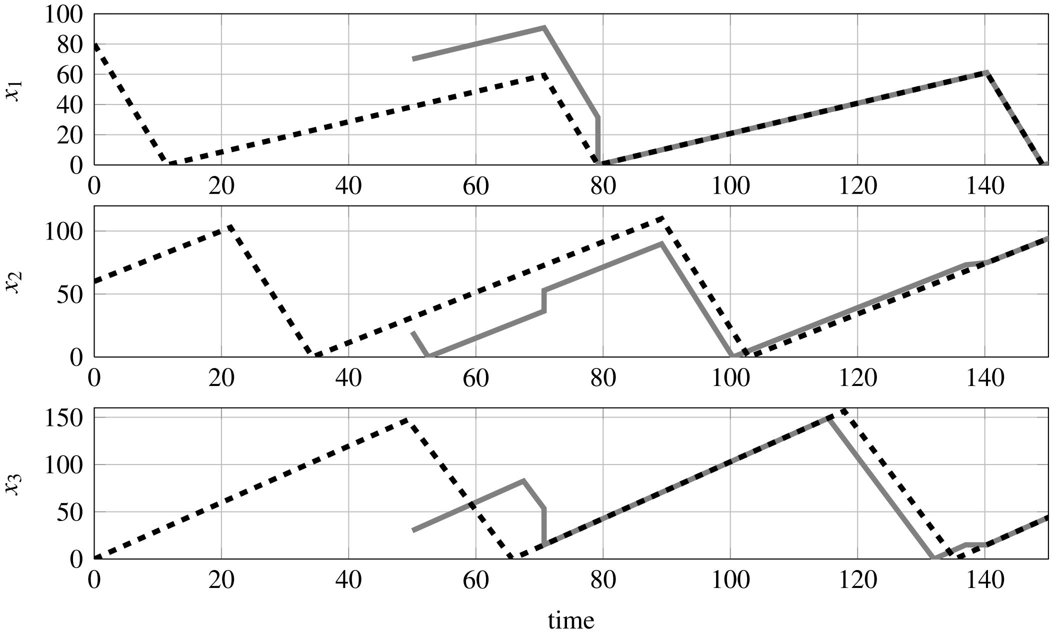


Figure 7.5: Actual (dashed lines) and estimated (solid lines) queue lengths of the 3-flow T-junction.

a re-entrant flow. The queues are served in the order from 1–4 after which the cycle is repeated. Furthermore, the system uses a clearing policy. The only measurable output of the system is the service rate of queue 2.

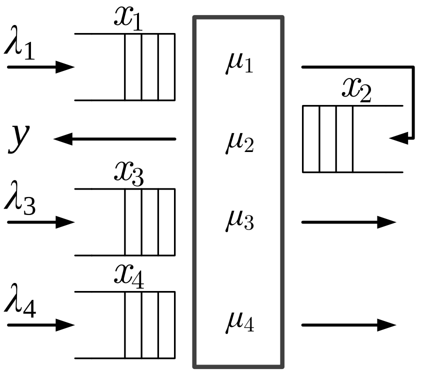


Figure 7.6: Layout of 4-queue switching server.

For the system with parameters

$$\lambda = \begin{bmatrix}1 & 0 & 3 & 4\end{bmatrix}^\top,$$
$$\mu = \begin{bmatrix}10 & 12 & 15 & 18\end{bmatrix}^\top,$$
$$\sigma = \begin{bmatrix}5 & 10 & 15 & 20\end{bmatrix}^\top,$$

and initial estimated state $\hat{x}(50) = \begin{bmatrix} 0 & 20 & 40 & 80 & 60 \end{bmatrix}^\top$, the actual and observed queue contents are presented in Figure 7.7. At the first visible event $t_1^v = 101.4$ (start serving queue 2), the observer state is updated. Since the estimated queue contents are lower than the actual queue contents, a waiting mode ($\hat{m} = 3\frac{1}{2}$) occurs during $t \in [214.6, 228.7]$. For the same reasoning, the observer is in waiting mode ($\hat{m} = 3\frac{1}{2}$) during $t \in [354.5, 355.8]$. Finally, at $t_5^v = 366.4$, the estimated state has converged to the actual state.

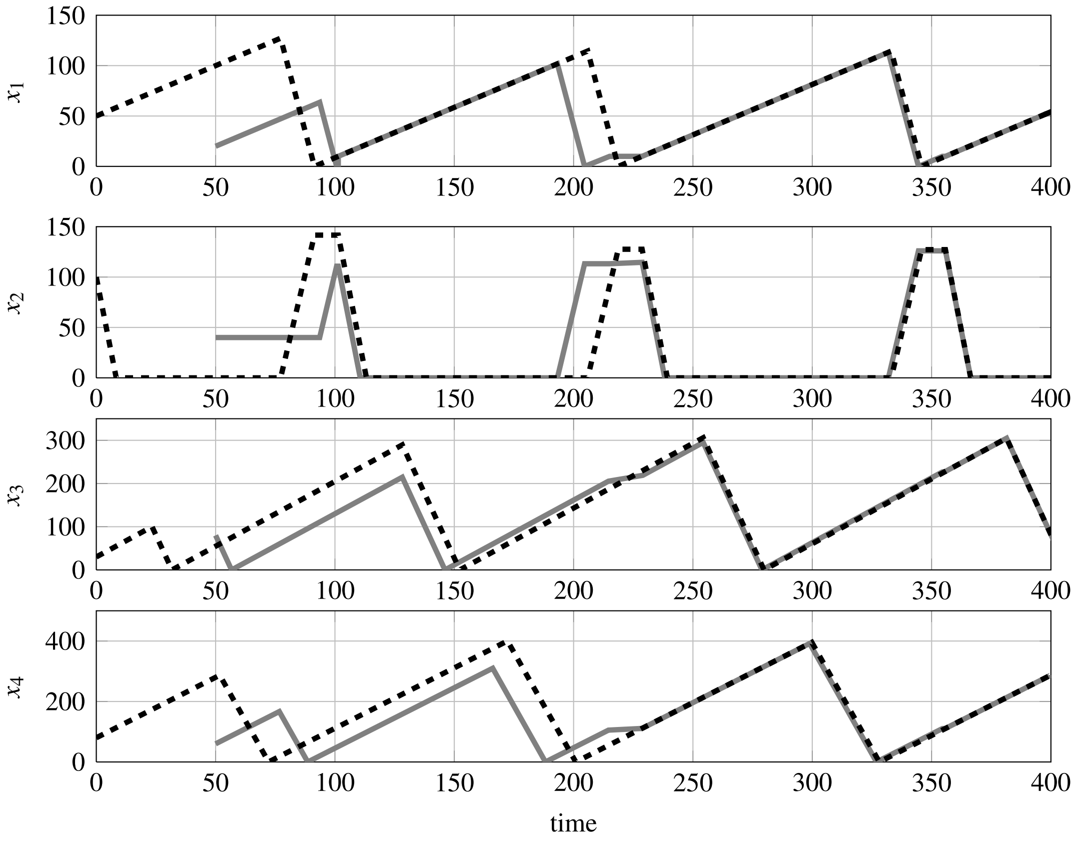


Figure 7.7: Actual (dashed lines) and estimated (solid lines) queue lengths of the 4-queue switching server.

7.5 Summary

This chapter presented a methodology to design observers for multi-queue switching servers which uses a clearing policy. Moreover, these systems, being highly relevant in the context of manufacturing and traffic applications, are part of a special class of piecewise affine hybrid systems (PWAHS). Although all subsystems are unobservable and not all events are visible, a continuous-time observer was constructed which guarantees that the estimated state converges to the actual state of the system under suitable conditions.

One of the main ideas in the construction of the continuous-time observer was sampling of the system at the visible events leading to a discrete-time periodic linear system, for which an observer can be designed using standard techniques from control theory. If this step is successfully performed, a continuous-time observer that asymptotically recovers the true state of the original hybrid systems can be synthesized. Indeed, the discrete-time observer can then be used as a blueprint for the continuous-time observer, where besides the plant dynamics additional 'waiting' modes are assigned to the observer. Occurrence of visible events before the time that the event was predicted to occur results in a discrete state switch and an update of the continuous states. The observer switches to a 'waiting' mode if an event occurs later than predicted. These principles are formally shown to result in a successful observer design. Via illustrations of a three-way traffic intersection and a 4-queue re-entrant switching server, the observer design and performance is presented. Important insights have been obtained, such as sampling at visible event times or the use of waiting modes, that might be fruitfully exploited into various research directions.

In the next chapter, observer design is extended towards networks of switching servers. For the Kumar Seidman network and a specific service policy, an observer is designed that reconstructs the state of the system. For the network, a visible event does not imply that the mode of the system is known, which is in contrast to the visible events for a single server system. Also, in Section 9.3.5, observer design for supply networks is presented.

Chapter 8

Observer design for the Kumar Seidman network

Observer design for a multi-queue single switching server with a clearing policy has been presented in Chapter 7. Some insights gained from this observer design are exploited in the current chapter to design an observer for a special two server network, i.e., the Kumar Seidman network using a specific policy. This is a first attempt in the direction of observer design for networks of switched systems.

In this chapter we focus on observer design for a well known network of switching servers, the Kumar Seidman network, which is also discussed in Chapter 5 regarding optimal periodic behavior. For this network, a control policy steering the network towards the desired periodic behavior is presented in [68]. In this chapter, an observer is derived that estimates the state of the network with the control policy from [68]. Given this control policy, the considered network is autonomous with linear dynamics, generates a constant output in each mode and the mode transitions are state-dependent. Only at some switches between modes the output reveals a minimal amount of information about the network, i.e., it reveals a set of possible modes. Furthermore, the switching pattern, or order of traversing modes, is not predefined, i.e., it is sometimes possible to switch to multiple modes. This complicates the observer design process compared to the process presented in Chapter 7, as for single server systems the measured output exactly indicates the current mode of the system and the order of switching between modes is fixed.

An approach to reconstruct the continuous and discrete state of the network with the control policy from [68] is presented, by measuring the output of the network. The approach is threefold. First, based on the policy and network dynamics, the switching pattern, i.e., the order in which the modes are traversed, is determined. Unlike the switching pattern for the single switching servers, see Chapter 7, this pattern is not fixed. However, this pattern is cyclic, which is exploited. Second, the network is sampled with varying sampling periods at so-called visible event times, i.e., times at which the output changes, and the dynamics between these times are derived. Third, the observer evaluates all possible predicted states at the visible event times. By eliminating unfeasible possibilities, it is shown that a single estimated state remains, which is equal to the actual network state.

The remainder of this chapter is organized as follows. Section 8.1 presents the Kumar Seidman network and the implemented control policy. The observer design is presented in Section 8.2. An example of the state reconstruction is presented in Section 8.3 and a summary is provided in Section 8.4.

8.1 System description

In this section the considered network is briefly discussed and the network policy is introduced, along with the problem formulation.

Kumar Seidman network

The Kumar Seidman network is depicted in Figure 8.1, which consists of two switching servers A and B. The fluid content of queue $n = 1,2,3,4$ at time t is denoted by $x_n(t)$. The observable output of the network is denoted by $y(t)$, which is the departure rate of queue 4. The service rate of queue n at time t is denoted by $r_n(t) \geq 0$. Switching of a server between serving queues i and j requires a setup time $\sigma_{i,j}$.

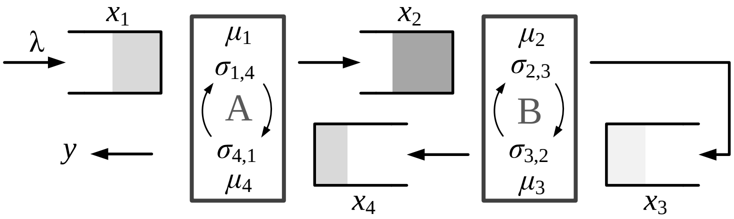


Figure 8.1: The switching server network introduced by Kumar and Seidman.

The state of this system $x(t)$ at time t is not only given by the queue contents, but also by the remaining setup time x_0^A at server A and the remaining setup time x_0^B at server B, i.e.,

$$x = \begin{bmatrix} x_0^A & x_0^B & x_1 & x_2 & x_3 & x_4 \end{bmatrix}^\top \in \mathbb{R}_+^{N+2},$$

with $N = 4$ being the number of queues and $\mathbb{R}_+ = [0,\infty)$. The mode of server j is denoted by m^j, $j = A,B$, and the mode of the system is given by $m = (m^A, m^B)$. For each queue n, server j has three server-dependent modes. Server mode $m^j = n$ indicates that server j is serving queue n at maximal rate μ_n, server mode $m^j = \underline{n}$ indicates that server j is serving queue n at arrival rate $(r < \mu_n)$, e.g., if $x_1 = 0$, and finally, server mode is $m^j = n^\sigma$ indicates that server j is setting up to serve queue n.

The dynamics of this system are hybrid. The continuous dynamics are described by

$$\dot{x}_0^A(t) = \begin{cases} -1 & \text{if } m^A \in \{1^\sigma, 4^\sigma\}, \\ 0 & \text{if } m^A \in \{1, 4\}, \end{cases} \qquad \dot{x}_0^B(t) = \begin{cases} -1 & \text{if } m^B \in \{2^\sigma, 3^\sigma\}, \\ 0 & \text{if } m^B \in \{2, 3\}, \end{cases} \tag{8.1a}$$

$$\dot{x}_1(t) = \lambda - r_1(t), \qquad \dot{x}_2(t) = r_1(t) - r_2(t), \tag{8.1b}$$

$$\dot{x}_4(t) = r_3(t) - r_4(t), \qquad \dot{x}_3(t) = r_2(t) - r_3(t). \tag{8.1c}$$

The servers in this network work at maximal rate, i.e., the service rate for queue n is the maximal rate μ_n if $x_n > 0$ or the service rate equals the arrival rate if $x_n = 0$, given that this rate does not exceed the maximal service rate. Therefore, at each time instant the service rates are subject to the constraints:

$$\begin{array}{llllr}
m^A \in \{1^\sigma, 4^\sigma\} & r_1 = 0 & r_4 = 0 & \text{for } x_0^A > 0, & (8.2\text{a}) \\
m^A = 1 & r_1 = \mu_1 & r_4 = 0 & \text{for } x_0^A = 0, x_1 > 0, & (8.2\text{b}) \\
m^A = \underline{1} & r_1 = \lambda & r_4 = 0 & \text{for } x_0^A = 0, x_1 = 0, & (8.2\text{c}) \\
m^A = 4 & r_1 = 0 & r_4 = \mu_4 & \text{for } x_0^A = 0, x_4 > 0, & (8.2\text{d}) \\
m^A = \underline{4} & r_1 = 0 & r_4 = \min(r_3, \mu_4) & \text{for } x_0^A = 0, x_4 = 0, & (8.2\text{e}) \\
m^B \in \{2^\sigma, 3^\sigma\} & r_2 = 0 & r_3 = 0 & \text{for } x_0^B > 0, & (8.2\text{f}) \\
m^B = 2 & r_2 = \mu_2 & r_3 = 0 & \text{for } x_0^B = 0, x_2 > 0, & (8.2\text{g}) \\
m^B = \underline{2} & r_2 = \min(r_1, \mu_2) & r_3 = 0 & \text{for } x_0^B = 0, x_2 = 0, & (8.2\text{h}) \\
m^B = 3 & r_2 = 0 & r_3 = \mu_3 & \text{for } x_0^B = 0, x_3 > 0, & (8.2\text{i}) \\
m^B = \underline{3} & r_2 = 0 & r_3 = 0 & \text{for } x_0^B = 0, x_3 = 0. & (8.2\text{j})
\end{array}$$

Constraints (8.2) describe that in case of a setup, no queues can be served and that a server, once serving a queue, serves that queue at maximal rate. Furthermore, we assume that service of a queue is only allowed after the server has finished the setup to that queue and that the network operates under ideal conditions, i.e., there are no server failures or breakdowns.

A transition in modes is triggered by an *event* ε, i.e., a switch between serving queues or a switch in service rates in a single server or in both servers. An event occurs if a specific queue in a mode reaches a threshold, such as, e.g., when the remaining setup time for the switch to serving queue 1 has expired, i.e., $x_0^A = 0$ and $m^A = 1^\sigma$.

The measurement information, which is the output of the network, is given by $y(t)$. Hence, as long as server A is in mode $m^A = 4$, this is directly seen in the output $y = \mu_4$ or if $m^A = \underline{4}$, this is seen in the output if $y = r_4 > 0$. When server A is in one of the other modes, the output y is equal to 0 and no information is available from the network. Note that in Chapter 7 the output, if measured, directly indicated the mode of the system. For the network in the current chapter, a measured output only indicates the state of server A. Hence, the state of the entire network is not necessarily known.

In the remainder of this chapter we use the following notations in which we make a distinction between visible and invisible modes, and visible and invisible events. A mode m is called visible if it generates an observable output $y = r_4 > 0$, i.e., $m^A = 4$ or $m^A = \underline{4}$ and $r_4 > 0$. Otherwise, a mode is called invisible. Transitions to and from visible modes are called *visible events* and transitions from invisible modes to invisible modes are called invisible events. Moreover, via the observable output, a distinction can be made between visible events leading to a visible mode, denoted by ε^v, or visible events leading to an invisible mode, denoted by ε^i. Hence, loosely speaking, we know when a visible event occurs and when the system enters or leaves a visible mode (which visible mode is not necessarily known).

Control policy

The control policy for this network, presented in [68] is summarized below.

- If in $m^A \in \{1,\underline{1}\}$ and $m^B \in \{3,\underline{3}\}$: switch to mode $(1,2)$. (8.3a)
- In $m^A \in \{1,\underline{1}\}$ and $m^B \in \{2,\underline{2}\}$: switch to mode (4,2) when $x_1 = 0$. (8.3b)
- In $m^A \in \{4,\underline{4}\}$ and $m^B \in \{2,\underline{2}\}$: switch to mode (4,3) when $x_2 = 0$ and $x_4 \leq \beta$. (8.3c)
- In $m^A \in \{4,\underline{4}\}$ and $m^B \in \{3,\underline{3}\}$): switch to mode (1,2) when $x_3 = 0$. (8.3d)

Given the network dynamics (8.1) and the policy (8.3), a cyclic evolution of events can be distinguished, which is presented in Figure 8.2. This cyclic mode evolution is required by the observer, as discussed at the end of this section. Modes are indicated by the circles and the transition between modes, triggered by an event, are indicated by arrows. For ease of reading, the brackets representing the modes are omitted in the figure, i.e., $(1,2) = 1,2$. The transitions that are triggered from visible events and the visible modes are represented by dashed lines. Note that in modes $(\underline{4},2)$ and $(\underline{4},3^\sigma)$ the output of the system is zero, as server A serves queue 4 at the arrival rate which is zero, and these modes are therefore invisible. Under normal operation, queue content $x_2 = 0$ at mode $(1^\sigma,2^\sigma)$. Hence, the successive mode of mode $(1^\sigma,2^\sigma)$ is $(1^\sigma,\underline{2})$ if $\sigma_{4,1} > \sigma_{3,2}$ and the successive mode of mode $(\underline{1},2^\sigma)$ is $(4^\sigma,\underline{2})$ if $\sigma_{4,1} < \sigma_{3,2}$.

Remark 8.1.1. *Based on the parameters of the system, some modes can be excluded from the mode evolution depicted in Figure 8.2. Only if $\mu_1 < \mu_2$, the mode $m = (1,\underline{2})$ exists and $m = (\underline{4},3)$ only exists if $\mu_3 < \mu_4$. Moreover, $m = (1,2^\sigma)$ and $m = (\underline{1},2^\sigma)$ are not possible if $\sigma_{4,1} > \sigma_{3,2}$ and $m = (1^\sigma,2)$ and $m = (1^\sigma,\underline{2})$ if $\sigma_{4,1} < \sigma_{3,2}$.*

The visible events leading to and from $m = (\underline{4},3)$ are unique, i.e., each transition identifies the current mode of the system. The visible event given by a transition in output from 0 to rate μ_3 indicates the visible event that triggers the transition from mode $(\underline{4},3^\sigma)$ to mode $(\underline{4},3)$. A change in output from μ_4 to μ_3 indicates the event triggering the transition between modes $(4,3)$ and $(\underline{4},3)$ and a change in output from μ_3 tot 0 indicates the transition between modes $(\underline{4},3)$ and $(1^\sigma,2^\sigma)$. These observable changes in output occur only at the aforementioned transitions. Hence, the modes following this transition can be directly observed. For all other visible events, no unique distinction is possible.

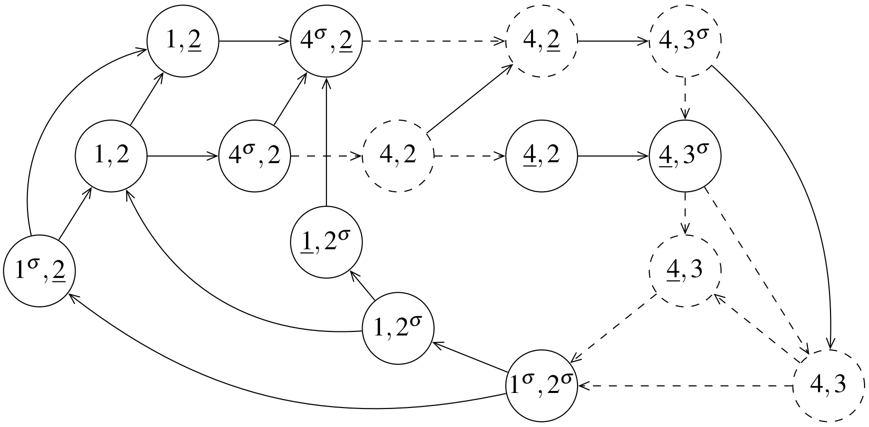


Figure 8.2: Cyclic evolution of modes. Dashed lines indicate visible modes and transitions triggered by visible events.

In the remainder of this chapter, we consider the parameters presented in [68], i.e.,

$$\lambda = 1, \qquad \beta = \tfrac{250}{3}, \tag{8.4a}$$

$$\sigma_{1,4} = \sigma_{4,1} = \sigma_{2,3} = \sigma_{3,2} = 50, \tag{8.4b}$$

$$\mu_1 = \mu_3 = \tfrac{1}{0.3}, \quad \mu_2 = \mu_4 = \tfrac{1}{0.6}. \tag{8.4c}$$

Since $\mu_1 > \mu_2$, $\mu_3 > \mu_4$ and all setups times are identical, some modes in Figure 8.2 can not be reached, as mentioned in Remark 8.1.1. The mode evolution for the system with parameters (8.4) is depicted in Figure 8.3. All events, visible and invisible, are labeled. To ensure that event ε_2^v occurs, $x_2 \geq 83\frac{1}{3}$ is required at the start of mode $(4^\sigma,2)$. Otherwise, while in mode $(4^\sigma,2)$, a switch to mode $(4^\sigma,\underline{2})$ occurs when $x_2 = 0$. However, this transition is discarded as it requires $x_1 < 166\frac{2}{3}$ at the start of mode $(1,2)$, which is impossible at normal operation, i.e., queue 1 is only served at mode $(1,2)$ (until it is empty) and receives a constant arrival rate $\lambda = 1$, between service of queue 1 a total of 150 time units is spend on setups and the remaining required $16\frac{2}{3}$ time units is spend in other modes. Moreover, as $\sigma_{4,1} = \sigma_{3,2}$, a direct transition from mode $(1^\sigma,2^\sigma)$ tot mode $(1,2)$ is possible. Also, mode $(1,\underline{2})$ does not exist, since $x_2 > 0$ as $\mu_1 > \mu_2$. For similar reasons, $m = (\underline{4},3)$ can not be reached, since $x_4 > 0$ as $\mu_3 > \mu_4$ and herewith also the events that uniquely identify the current mode of the system are not present. Furthermore, the service rate at server A for mode $\underline{4}$ is zero when $m^B \neq 3$, hence the modes $(\underline{4},2)$ and $(4,3^\sigma)$ are unobservable. For $m = (\underline{1},\underline{2})$ the service rate at both queues is λ.

The discrete dynamics for the network with policy (8.3) and parameters (8.4), along

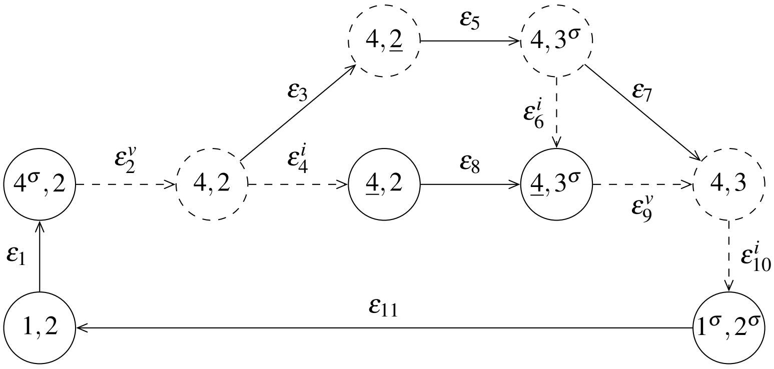


Figure 8.3: Cyclic evolution of modes for the system with parameters (8.4). Dashed lines indicate visible modes and transitions triggered by visible events.

with the event labels, are as follows,

$$
\begin{array}{llll}
m := (4^\sigma,2) \wedge x_0 := \sigma_{1,4} & \text{if } m = (1,2) \wedge x_1 = 0 & (\text{event } \varepsilon_1), & (8.5a)\\
m := (4,2) & \text{if } m = (4^\sigma,2) \wedge x_0 = 0 & (\text{event } \varepsilon_2^v), & (8.5b)\\
m := (4,\underline{2}) & \text{if } m = (4,2) \wedge x_2 = 0 & (\text{event } \varepsilon_3), & (8.5c)\\
m := (\underline{4},2) & \text{if } m = (4,2) \wedge x_4 = 0 & (\text{event } \varepsilon_4^i), & (8.5d)\\
m := (4,3^\sigma) \wedge x_0 := \sigma_{2,3} & \text{if } m = (4,\underline{2}) \wedge x_4 \leq \beta & (\text{event } \varepsilon_5), & (8.5e)\\
m := (\underline{4},3^\sigma) & \text{if } m = (4,3^\sigma) \wedge x_4 = 0 & (\text{event } \varepsilon_6^i), & (8.5f)\\
m := (4,3) & \text{if } m = (4,3^\sigma) \wedge x_0 = 0 & (\text{event } \varepsilon_7), & (8.5g)\\
m := (\underline{4},3^\sigma) \wedge x_0 := \sigma_{2,3} & \text{if } m = (\underline{4},2) \wedge x_2 = 0 & (\text{event } \varepsilon_8), & (8.5h)\\
m := (4,3) & \text{if } m = (\underline{4},3^\sigma) \wedge x_0 = 0 & (\text{event } \varepsilon_9^v), & (8.5i)\\
m := (1^\sigma,2^\sigma) \wedge x_0 := \sigma_{4,1} & \text{if } m = (4,3) \wedge x_3 = 0 & (\text{event } \varepsilon_{10}^i), & (8.5j)\\
m := (1,2) & \text{if } m = (\underline{4},3^\sigma) \wedge x_0 = 0 & (\text{event } \varepsilon_{11}). & (8.5k)
\end{array}
$$

The cyclic evolution of modes is a requirement for the observer design, as it bounds the number of possible events and number of possible modes. Furthermore, it is also required that the visible events are created by an unique trajectory of the system, i.e., if multiple trajectories create identical visible events, the observer is not able to distinguish between the trajectories. From the cyclic evolution of modes, we make the following observations. It can be seen in Figure 8.3 that between modes $(4,2)$ and $(4,3)$ multiple routes are possible, listed below

$$
\begin{array}{llll}
\text{Route 1:} & (4,2) \to (4,\underline{2}) \to (4,3^\sigma) \to (4,3) & \text{if } \beta + x_2^{(4,2)} & < x_4^{(4,2)}\\
\text{Route 2:} & (4,2) \to (4,\underline{2}) \to (4,3^\sigma) \to (\underline{4},3^\sigma) \to (4,3) & \text{if } \quad x_4^{(4,2)} & \leq \beta + x_2^{(4,2)}\\
 & (4,2) \to (\underline{4},2) \to (\underline{4},3^\sigma) \to (4,3), & &
\end{array}
$$

where x_n^m denotes the content of queue n at the start of mode m. Route 2 contains two different trajectories, the state at the start of mode $(4,3)$ is however identical in both trajectories, as both queues 2 and 4 are emptied. The duration of route 1 is given by $\frac{x_4^{(4,2)}}{\mu_4}$ and the duration of route 2 equals $\frac{x_2^{(4,2)}}{\mu_2}+\sigma_{2,3}$. Then, by regarding the system dynamics in the traversed modes, the transition maps from the start of mode $(4,3)$ in cycle k to the start of mode $(4,3)$ in the successive cycle $(k+1)$ are given by

$$x^{(4,2)}(k+1)=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&\frac{5}{7}&\frac{3}{14}&\frac{3}{14}&\frac{3}{7}\\0&0&\frac{5}{7}&\frac{3}{14}&\frac{3}{14}&\frac{3}{7}\\0&0&0&\frac{1}{2}&\frac{1}{2}&0\end{bmatrix}x^{(4,2)}(k)+\begin{bmatrix}0\\0\\50\\-\frac{1000}{21}\\\frac{2500}{21}\\0\end{bmatrix}\quad \text{if } x_4^{(4,2)}(k)>\beta+x_2^{(4,2)}(k), \tag{8.6a}$$

$$x^{(4,2)}(k+1)=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&\frac{5}{7}&\frac{9}{14}&\frac{3}{14}&0\\0&0&\frac{5}{7}&\frac{9}{14}&\frac{3}{14}&0\\0&0&0&\frac{1}{2}&\frac{1}{2}&0\end{bmatrix}x^{(4,2)}(k)+\begin{bmatrix}0\\0\\50\\-\frac{250}{21}\\\frac{3250}{21}\\0\end{bmatrix}\quad \text{if } x_4^{(4,2)}(k)\leq\beta+x_2^{(4,2)}(k). \tag{8.6b}$$

Using (8.6), it can be shown that, for the system with initial mode $(4,2)$, after three cycles the system does not switch anymore between routes, i.e., the system either follows route 1 or route 2, and the system eventually converges to state

$$x^{(4,2)}(\infty)=\begin{bmatrix}0 & 0 & 50 & 416\frac{2}{3} & 583\frac{1}{3} & 500\end{bmatrix}^\top,$$

which is a part of the desired periodic behavior given in [68]. Then, the time between visible events are unique, i.e., there exists only a single trajectory generating these events. This completes the formulation of the model and the policy. Below we formulate the problem of reconstructing the network's state.

Problem Formulation

For the deterministic fluid flow network presented in Figure 8.1 with parameters (8.4) and policy (8.3), reconstruct the network's state based on the measurement information $y(t)$.

8.2 Observer design

Notice that information about the network's state is only received when visible events occur. Furthermore, as can be seen in Figure 8.3, observing a visible event that leads to a visible mode does not include enough information to distinguish between events ε_2^v or ε_9^v. Likewise, for visible events that lead to an invisible mode, no distinction is possible between events ε_4^i, ε_6^i or ε_{10}^i. To cope with this issue, the proposed observer estimates a set of all possible states and, by receiving additional information at visible events, eliminates infeasible possibilities until the entire network's state is reconstructed.

Extra system information, used to derive the observer, can be obtained from the cyclic mode evolution and discrete dynamics (8.5), i.e., some states are known at the start of each mode. For instance, at the start of mode $(1,2)$, x_0 is zero, since there are no setups. Also, $x_2 = x_3 = 0$, as these queues are emptied in, or before, arriving in mode $(4,3)$, and no fluid arrives at these queues thereafter. The queue contents at the start of each mode are presented in Table 8.1. If an $*$ is displayed, the queue content at the start of that mode is (initially) unknown.

Table 8.1: System state at start of mode m.

m	$x_0^{A,m}$	$x_0^{B,m}$	x_1^m	x_2^m	x_3^m	x_4^m
$(1,2)$	0	0	$*$	0	0	$*$
$(4^\sigma,2)$	$\sigma_{1,4}$	0	0	$*$	$*$	$*$
$(4,2)$	0	0	$\lambda\sigma_{1,4}$	$*$	$*$	$*$
$(4,\underline{2})$	0	0	$*$	0	$*$	$*$
$(4,3^\sigma)$	0	$\sigma_{2,3}$	0	$*$	$*$	$*$
$(\underline{4},2)$	0	0	$*$	$*$	$*$	0
$(\underline{4},3^\sigma)$	0	$[0,\sigma_{2,3}]$	$*$	0	$*$	0
$(4,3)$	0	0	$*$	0	$*$	0
$(1^\sigma,2^\sigma)$	$\sigma_{4,1}$	$\sigma_{3,2}$	$*$	0	0	$*$

With the network depicted in Figure 8.1 and policy (8.3), we derive an observer that, given the arrival rate and by measuring the output rate, estimates the current state of the system. To do so, first the dynamics between visible events are considered, and will be presented below.

Event dynamics

In a mode, the dynamics are linear and jumps in the state occur only in x_0, right after events leading to a mode with a setup. In the remainder, j is used as the visible event counter and visible event times are denoted by t_j^v. Moreover, let $\Delta t_j = t_{j+1}^v - t_j^v$ denote the time between visible event j and $j+1$. For visible event $j+1$, we determine the state of the system directly after the event, which is based on Δt_j. Also, the time it takes for the next event to occur is predicted, i.e., Δt_{j+1}. Since sometimes multiple routes between visible events are possible, multiple times to the next event can be predicted, e.g., after visible event ε_2^v the successive visible event can be either ε_4^i, ε_6^i or ε_{10}^i. Below, for each visible event, the state of the system directly after the event and the estimated time(s) to the next event(s) are presented.

Event ε_2^v

Consider event ε_2^v, i.e., the transition between modes $(4^\sigma,2)$ and $(4,2)$, that occurs at $t=t_{j+1}^v$. Then, from Table 8.1 it can be seen that $x_0^A(t_{j+1}^v)=x_0^B(t_{j+1}^v)=0$ and $x_1(t_{j+1}^v)=\lambda\sigma_{1,4}$. Furthermore, queue contents x_2 and x_3 are zero at the preceding visible event, i.e., event ε_{10}^i at $t=t_j^v$. Also, the time spent in modes $(1^\sigma,2^\sigma)$, $(1,2)$ and $(4^\sigma,2)$ can be derived, which are $\sigma_{4,1}$, $\Delta t_j-\sigma_{4,1}-\sigma_{1,4}$ and $\sigma_{1,4}$ respectively. Therefore, the contents at $t=t_{j+1}^v$ of queues 2 and 3 can be derived. Queue 2 increases with rate μ_1 during mode $(1,2)$ and decreases with rate μ_2 during modes $(1,2)$ and $(4^\sigma,2)$. The fluid that leaves queue 2 flows to queue 3. The content of queue 4 does not change between the visible events, as both queues 3 and 4 are not served. Hence, the state of the system directly after event ε_2^v is given by

$$\begin{aligned}
x_0^A(t_{j+1}^v)&=0,\\
x_0^B(t_{j+1}^v)&=0,\\
x_1(t_{j+1}^v)&=\lambda\sigma_{1,4},\\
x_2(t_{j+1}^v)&=\mu_1(\Delta t_j-\sigma_{1,4}-\sigma_{4,1})-\mu_2(\Delta t_j-\sigma_{1,4}),\\
x_3(t_{j+1}^v)&=\mu_2(\Delta t_j-\sigma_{1,4}),\\
x_4(t_{j+1}^v)&=x_4(t_j^v).
\end{aligned}$$

It can be seen that the content of queue 4 is the only unknown queue content in this case, as the remainder of the state is either known or can be derived from the time between the visible events. Furthermore, given the queue contents at time t_{j+1}^v, the duration Δt_{j+1} between the event $j+1$ and the successive visible event $j+2$ can be predicted. The successive visible event of ε_2^v can be either event ε_4^i, ε_6^i or ε_{10}^i, based on the state information. These events occur if

$$\begin{aligned}
&\text{event } \varepsilon_4^i, && \text{if } \Delta t_{j+1}=\frac{x_4}{\mu_4} \quad\wedge\quad \Delta t_{j+1}\le\frac{x_2}{\mu_2}, && (8.8a)\\
&\text{event } \varepsilon_6^i, && \text{if } \Delta t_{j+1}=\frac{x_4}{\mu_4} \quad\wedge\quad \Delta t_{j+1}\ge\frac{x_2}{\mu_2}, && (8.8b)\\
&\text{event } \varepsilon_{10}^i\text{via } \varepsilon_2^v, && \text{if } \Delta t_{j+1}=\frac{x_4}{\mu_4}+\frac{x_2+x_3}{\mu_3} \quad\wedge\quad \Delta t_{j+1}>\frac{x_2}{\mu_2}+\sigma_{2,3}, && (8.8c)
\end{aligned}$$

where, for ease of reading, $x_n(t_{j+1}^v)$ is denoted by x_n. Note that event ε_{10}^i can be reached via multiple routes, therefore we denoted in (8.8c) that ε_{10}^i occurs after visible event ε_2^v, which results in different dynamics as arriving via a route with visible event ε_9^v. Moreover, along with the estimated successive event, also extra information about the unknown queue content x_4 is derived. For (8.8a), we know that $x_4-x_2\le 0$, for (8.8b) we have $0\le x_4-x_2\le\mu_2\sigma_{2,3}$ and for (8.8c) we have $\mu_2\sigma_{2,3}\le x_4-x_2$. This information is used by the observer to determine the feasibility of the estimated state.

For the other visible events, the queue contents and successive estimated events are derived similarly and are listed below. Note that for events ε_9^v and ε_{10}^i, the preceding events are required to derive the dynamics, as multiple routes can lead to these events.

Event ε_4^i

$$\begin{aligned}
x_0^A(t_{j+1}^v) &= 0,\\
x_0^B(t_{j+1}^v) &= 0,\\
x_1(t_{j+1}^v) &= \lambda(\sigma_{1,4}+\Delta t_j),\\
x_2(t_{j+1}^v) &= x_2(t_j^v)-\mu_2\Delta t_j,\\
x_3(t_{j+1}^v) &= x_3(t_j^v)+\mu_2\Delta t_j,\\
x_4(t_{j+1}^v) &= 0.
\end{aligned}$$

$$\varepsilon_9^v \text{via } \varepsilon_4^i \text{ if } \Delta t_{j+1} = \frac{x_2}{\mu_2}+\sigma_{2,3}.$$

Event ε_6^i

$$\begin{aligned}
x_0^A(t_{j+1}^v) &= 0,\\
x_0^B(t_{j+1}^v) &\in [0,\sigma_{2,3}],\\
x_1(t_{j+1}^v) &= \lambda(\sigma_{1,4}+\Delta t_j),\\
x_2(t_{j+1}^v) &= 0,\\
x_3(t_{j+1}^v) &= x_2(t_j^v)+x_3(t_j^v),\\
x_4(t_{j+1}^v) &= 0.
\end{aligned}$$

$$\varepsilon_9^v \text{via } \varepsilon_9^v \text{ if } \Delta t_{j+1} \le \sigma_{2,3}.$$

Event ε_9^v, via ε_2^v

$$\begin{aligned}
x_0^A(t_{j+1}^v) &= 0,\\
x_0^B(t_{j+1}^v) &= 0,\\
x_1(t_{j+1}^v) &= x_1(t_j^v)+\lambda\Delta t_j,\\
x_2(t_{j+1}^v) &= 0,\\
x_3(t_{j+1}^v) &= x_3(t_j^v),\\
x_4(t_{j+1}^v) &= 0.
\end{aligned}$$

$$\varepsilon_{10}^i \text{via } (\underline{4},3^\sigma) \text{ if } \Delta t_{j+1} = \frac{x_3}{\mu_3}.$$

Event ε_9^v, via ε_4^i

$$\begin{aligned}
x_0^A(t_{j+1}^v) &= 0,\\
x_0^B(t_{j+1}^v) &= 0,\\
x_1(t_{j+1}^v) &= x_1(t_j^v)+\lambda\Delta t_j,\\
x_2(t_{j+1}^v) &= 0,\\
x_3(t_{j+1}^v) &= x_3(t_j^v)+\mu_2(\Delta t_j-\sigma_{2,3}),\\
x_4(t_{j+1}^v) &= 0.
\end{aligned}$$

$$\varepsilon_{10}^i \text{via } (\underline{4},3^\sigma) \text{ if } \Delta t_{j+1} = \frac{x_3}{\mu_3}.$$

Event ε_{10}^i, via ε_2^v

$$\begin{aligned}
x_0^A(t_{j+1}^v) &= \sigma_{4,1},\\
x_0^B(t_{j+1}^v) &= \sigma_{3,2},\\
x_1(t_{j+1}^v) &= x_1(t_j^v)+\lambda\Delta t_j,\\
x_2(t_{j+1}^v) &= 0,\\
x_3(t_{j+1}^v) &= 0,\\
x_4(t_{j+1}^v) &= \frac{\mu_3-\mu_4}{\mu_3}(x_2(t_j^v)+x_3(t_j^v)).
\end{aligned}$$

$$\varepsilon_2^v \text{if } \Delta t_{j+1} = \frac{x_1+\lambda\sigma_{4,1}}{\mu_1-\lambda}+\sigma_{4,1}+\sigma_{1,4}.$$

Event ε_{10}^i, via ε_9^v

$$\begin{aligned}
x_0^A(t_{j+1}^v) &= \sigma_{4,1},\\
x_0^B(t_{j+1}^v) &= \sigma_{3,2},\\
x_1(t_{j+1}^v) &= x_1(t_j^v)+\lambda\Delta t_j,\\
x_2(t_{j+1}^v) &= 0,\\
x_3(t_{j+1}^v) &= 0,\\
x_4(t_{j+1}^v) &= (\mu_3-\mu_4)\Delta t_j.
\end{aligned}$$

$$\varepsilon_2^v \text{if } \Delta t_{j+1} = \frac{x_1+\lambda\sigma_{4,1}}{\mu_1-\lambda}+\sigma_{4,1}+\sigma_{1,4}.$$

Observer

Given the dynamics of the network between visible events and the estimated times to a successive visible event presented above, the observer is designed. At the initial visible event, multiple options for the system's state are possible, i.e., the system is in one of the modes $(4,2)$ or $(4,3)$ when the visible event leads to a visible mode (ε^v) and the system is in one of the modes $(\underline{4},2)$, $(\underline{4},3^\sigma)$ and $(1^\sigma,2^\sigma)$ otherwise (ε^i), see Figure 8.3. An observed variable i is indicated by $\hat{i}$. In the remainder, we denote by $\hat{\chi}(t_i)$ the set of estimated states at event i. An estimated state is given by

$$\begin{bmatrix}\hat{x}_0^A & \hat{x}_0^B & \hat{x}_1 & \hat{x}_2 & \hat{x}_3 & \hat{x}_4 & \hat{m} & \Delta\hat{t}_{j+1}\end{bmatrix}^\top,$$

i.e., the estimated system state, estimated mode and estimated time(s) to the next event. By evaluating the estimated time to the next event $\Delta\hat{t}_{j+1}$, based on the (partially known) state of the system, and by comparing this time to the actual occurrence of the next event, the observer's states are updated. The number of possible states can increase, when a specific queue content required to determine the time to the next event is unknown, e.g., if x_4 is unknown after event ε_2^v multiple mode routes are possible, see (8.8). The number of possible states decreases when a possible state becomes infeasible, i.e., when the estimated time to the next event does not match the actual time to the next event or at negative estimated queue contents. The evaluation of estimated states at the visible events continues until a single estimated state remains. Then, this state is the estimated state and is equal to the network's state, as shown by experiments below.

To ensure that the observer eventually ends up with a single state estimation, we evaluate the performance of the observer, by initiating the network at mode $(1,2)$, as for all possible initial states the network will (eventually) be in this mode due to the cyclic behavior. The network initiates in mode $(1,2)$ with $x_0^A = x_0^B = x_2 = x_3 = 0$, see Table 8.1, and by evaluating all possible initial contents of queues 1 and 4, i.e., $x_1(0) \geq 166\frac{2}{3}$ and $x_4(0) \geq 0$, the number of visible events required by the observer to end up at a single estimated state are counted. The results of these simulations are presented in Figure 8.4. It can be seen that after maximally six visible events the observer ends up with a single state estimate. Moreover, once a single state estimate remains, the estimated state has also converged to the actual state of the system. Also, for larger initial queue contents, which are not depicted in the figure, the observer ends with a single estimated state. This suggests that the proposed observer works as desired. However, no rigorous proof is provided. To gain more insight in the observer, an illustration of the state estimation process is presented next.

8.3 Illustration

Consider the network with initial queue contents at $t = 0$ given by

$$x(0) = \begin{bmatrix}50 & 50 & 650 & 0 & 0 & 500\end{bmatrix}^\top$$

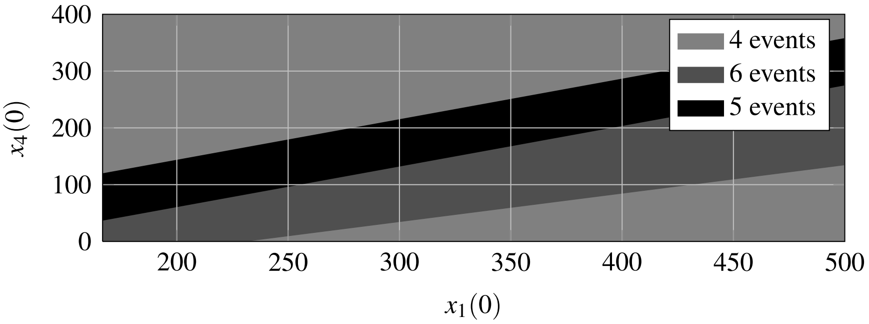


Figure 8.4: Number of visible events required by the observer before a single solution remains, starting in mode $(1,2)$ and $x_0^A = x_0^B = x_2 = x_3 = 0$.

and initial mode $m(0) = (1^\sigma, 2^\sigma)$, which is part of the desired periodic behavior described in [68]. The queue content evolution is presented in Figure 8.5. The upper figure presents the queue contents of server A, the figure in the middle the queue contents of server B and the measured output is presented in the figure at the bottom.

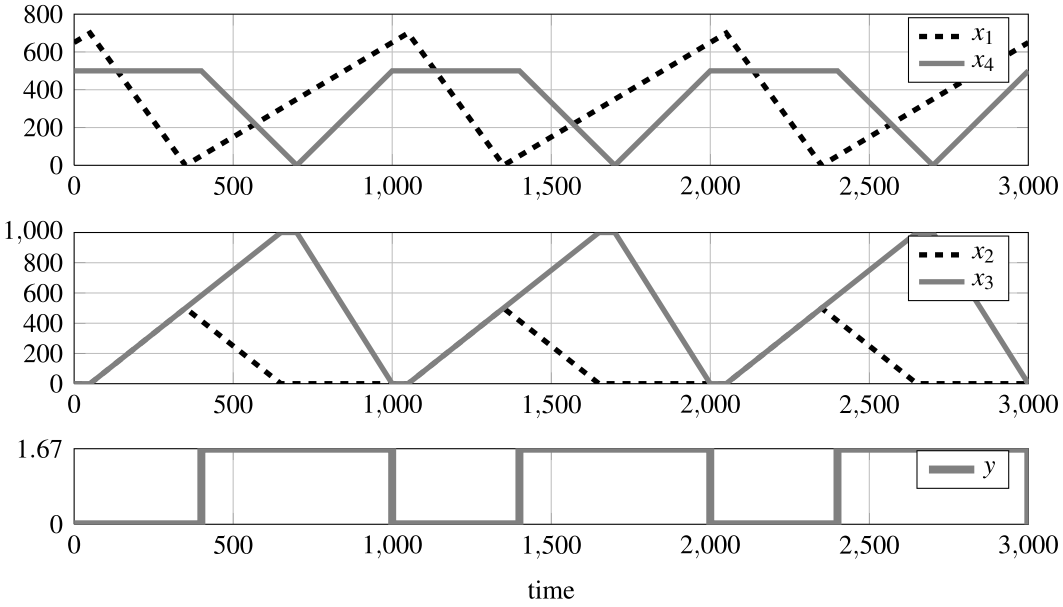


Figure 8.5: Queue contents and output for the network with $m(0) = (1^\sigma, 2^\sigma)$ and $x(0) = [50, 50, 650, 0, 0, 500]^\top$.

The observer starts to estimate the state of the system at time of the first visible event, i.e., at $t = t_1^v = 400$ (event ε_2^v at the actual network). This event leads to a visible mode, therefore, the set of estimated modes is $\hat{m}(t_1^v) = \{(4,2),(4,3)\}$. Mode $(4,3)$ can be reached in two ways, via visible event ε_2^v or visible event ε_9^v. However, at t_1^v the queue contents are unknown, hence these estimated states are identical.

Then the set of estimated states at the first visible event is given by:

$$\hat{\chi}(t_1^v) = \left\{ \begin{bmatrix} 0 \\ 0 \\ 50 \\ * \\ * \\ * \\ (4,2) \\ * \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ * \\ 0 \\ * \\ 0 \\ (4,3) \\ \geq 121.4 \end{bmatrix} \right\}. \tag{8.15}$$

It can be seen that, despite the contents of many queues are unknown, a lower bound of the estimate for the time to the next event can be derived. The second visible event ε_{10}^i is measured at $t_2^v = 1000$, hence $\Delta t_1 = 600$ and both estimated states at time of the first event are feasible. The corresponding estimated states at $t = t_2^v$ are given by

$$\hat{\chi}(t_2^v) = \left\{ \begin{bmatrix} 0 \\ 0 \\ 650 \\ * - 1000 \\ * + 1000 \\ 0 \\ (\underline{4},2) \\ * \end{bmatrix}, \begin{bmatrix} 0 \\ [0, \sigma_{2,3}] \\ 650 \\ 0 \\ * \\ 0 \\ (\underline{4},3^\sigma) \\ [0-50] \end{bmatrix}, \begin{bmatrix} 50 \\ 50 \\ 650 \\ 0 \\ 0 \\ * \\ (1^\sigma,2^\sigma) \\ 400 \end{bmatrix}, \begin{bmatrix} 50 \\ 50 \\ * + 600 \\ 0 \\ 0 \\ 1000 \\ (1^\sigma,2^\sigma) \\ \geq 357 \end{bmatrix} \right\}. \tag{8.16}$$

Note that the number of estimated states has grown to four possible states, as after event ε_2^v three different successive visible events are possible. At $t_3^v = 1400$, the third event is observed. The second estimated state of $\hat{\chi}(t_2^v)$ does not satisfy $\Delta t_2 = 400$, and is therefore discarded. Then, the estimated states at $t = t_3^v$ are as follows

$$\hat{\chi}(t_3^v) = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1050 \\ 0 \\ * + 1583\frac{1}{3} \\ 0 \\ (4,3)((\underline{4},2)) \\ \geq 475 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 50 \\ 416\frac{2}{3} \\ 583\frac{1}{3} \\ * \\ (4,2) \\ \leq 250, > 250, \geq 300 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 50 \\ 416\frac{2}{3} \\ 583\frac{1}{3} \\ 1000 \\ (4,2) \\ 600, 900 \end{bmatrix} \right\}. \tag{8.17}$$

As in mode $(4,2)$ multiple options for the next event are possible, see (8.8), multiple times are presented for the final two estimated states. At $t_4^v = 2000$, the fourth event is measured, $\Delta t_4 = 600$, this eliminates two estimated states. Right after this event, the estimated state are

$$\hat{\chi}(t_4^v) = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1650 \\ 0 \\ 0 \\ 1000 \\ (4,3)((\underline{4},3^\sigma)) \\ 826.6 \end{bmatrix}, \begin{bmatrix} 0 \\ [0,\sigma_{2,3}] \\ 650 \\ 0 \\ 1000 \\ 0 \\ (\underline{4},3^\sigma) \\ \leq 50 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 650 \\ 0 \\ 0 \\ 500 \\ (1^\sigma,2^\sigma) \\ 400 \end{bmatrix} \right\}. \tag{8.18}$$

Finally, at $t = 2400$ the fifth visible event is measured. Then, since $\Delta t_5 = 400$, a single state set remains, for which the estimated states have converged to the actual states.

8.4 Summary

In this chapter, a methodology is presented for observer design for the Kumar Seidman network with a given and known policy. This chapter elaborated on the single switching server observer design method presented in Chapter 7. The network observer design approach is a first step towards observer design for fluid flow networks of switching servers. Although a minimal amount of information is measured, an observer can be derived which converges to the current network state. Given the network topology and the policy, a cyclic evolution of modes, and therefore also events, has been determined. Furthermore, the policy and used parameters resulted in a system for which each trajectory has unique visible events, which is also a required for the observer design. By deriving the dynamics between observable (visible) events, an observer has been derived which estimates a set of all possible states and, by receiving additional information at visible events, eliminates infeasible estimated state sets until the entire state of the network is reconstructed. Via simulation it is shown that for a wide range of initial states, the observer results in a single estimated state and converges to both the discrete and continuous state of the network after at most six visible events. This suggests that the proposed observer works as desired. However, no rigorous proof of the state convergence is provided.

Chapter 9

Related Problems

Control and observer design of systems of switching servers have been presented in the previous chapters. In this chapter, some related problems are discussed. These problems are presented in two categories: traffic intersections and supply networks.

For vehicles in a network of signalized traffic intersections, the transportation time (travel time) between intersections is non-negligible, as the vehicles require to cover some distance to arrive at the next intersection. Therefore, the assumption on instant transportation in Chapter 5 is not valid for a network of traffic intersections. Therefore, a heuristic based on optimal behavior of single multi-queue switching servers is presented in Section 9.1 that provides good, not necessarily optimal, schedules for networks of traffic intersections. This heuristic is based on the optimal periodic behavior of single multi-queue switching servers, as presented in Chapter 4. Furthermore, in Section 9.2, we present an approach to derive service schedules for traffic intersections where the time between arrivals is stochastic.

In Section 9.3, optimal control and observer design for a class of supply networks are presented. The control strategy to organize the flow of goods is extremely relevant in managing supply networks. Handling of uncertainty, e.g. customer demand, is often a considerable challenge. We introduce methods of constrained robust optimal control, a technique from control theory, to compute the explicit control strategy for small-sized demand-driven discrete-time controlled dynamical systems with uncertainties. This allows avoiding any assumptions about the form of the policy a priori. The aim is to show the applicability of the methods, instead of finding innovative policies, as the resulting policies are well known. Another control problem arises when control actions depend on the information of multiple states and these states are not (completely) known, e.g. due to communication issues. Here observers are used, if possible, to derive the required state information based on the in- and output of the network. By means of an example, it is shown that by using these observers, the control policy can be perfectly utilized and no longer depends on communication.

9.1 Network with transportation times

In Chapter 4, networks with no or negligible transportation times are considered, and for these networks the periodic behavior can be derived. However, regarding networks of traffic intersections for example, vehicles travel between the intersections and this requires travel (or transportation) times which can not be neglected. Therefore, a heuristic to derive a good schedule for networks with transportation, based on optimal behavior of single multi-queue switching servers, is presented below.

As illustration, consider the network of two signalized traffic intersections, depicted in Figure 9.1. The network consists of four routes and six queues. The routes through queues 3 and 4 travel perpendicular to the other two routes. Once the vehicles in queue 1 are served, they travel to queue 2 at the other intersection. Total transportation time from leaving the first intersection and arriving at the queue of the next intersection is denoted by γ_1. The same holds for vehicles that travel from queue 5 to queue 6 with a transportation time denoted by γ_2. Note that the transportation times γ_1 and γ_2 can be different, e.g., due to the layout of the intersection or road quality.

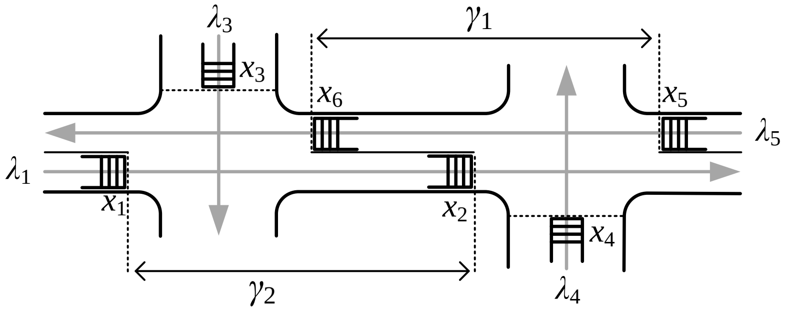


Figure 9.1: Two intersection network with four flows.

For a network where traffic is allowed to flow between two intersections in a single direction via a single route, the optimal periodic behavior can be derived using optimal behavior of a single intersections. For instance, consider the network in Figure 9.1 without the route leading through queues 5 and 6. In this case, vehicles can only travel between intersections in one direction, i.e., from queue 1 to queue 2. Then, given the green period of queue 1, it is possible to set the green period of queue 2 such that it stays empty. The optimal service periods for the queues 1, 3 and 4, for the system without transportation times, can be derived by the method presented in Chapter 4, i.e., by neglecting the transportation time the network can be seen as a single intersection (without queue 2, and treating queues 1 and 4 as conflicting). Next, we take the service period of queue 2 identical to the derived service period of queue 1. Then, this schedule with the start of the service period of queues 2 and 4 delayed with γ_1, with respect to queues 1 and 3 of the system without transportation times, yields the optimal schedule for this system with transportation times. For this schedule, vehicles always receive a green light when arriving at queue 2, hence $x_2(t) = 0$. The delay between initiating the service periods of queues 1 and 2 is referred to as the *phase delay*, denoted by ξ. So, for the system with queues 1-4, the phase delay equals the transportation time $\xi = \gamma_1$.

If vehicles flow in both directions between intersections in the network, i.e., considering all routes in Figure 9.1, this approach does not work anymore, as it is impossible to achieve empty queues 2 and 6 for all possible transportation times $\gamma_1, \gamma_2 \geq 0$, given the optimal schedule for the system without transportation times. Note that for some specific time delays it is possible to find a phase delay that ensures both intermediate queues are zero. We propose a method based on the techniques of Chapter 4 to derive a service schedule. This is discussed below.

To derive a schedule using techniques from Chapter 4, the network depicted in Figure 9.1 can be abstracted to a single intersection by neglecting the transport between queues 1 and 2 and between queues 5 and 6. To ensure stability of the system, the arrival rates at queues 2 and 6 are chosen identical to the arrival rates of queues 1 and 5, respectively, as the amount of vehicles served in these intermediate queues are equal to the amount of vehicles served in the preceding queues. Moreover, queues 2 and 6 are relocated next to queues 3 and 4 respectively, in such a way that they are conflicting. This abstracted system is depicted in Figure 9.2.

For the abstracted system, the optimal service schedule can be easily derived. From this schedule, the schedule for the network with transportation times is derived in two steps. First, the service schedule is chosen identical to the service schedule of the system without transportation times. Second, service of queues at the intersection on the right (queues 2, 4 and 5) is delayed with phase delay ξ. The phase delay is chosen such that sum of the intermediate queue contents, i.e., queues 2 and 6, is minimal.

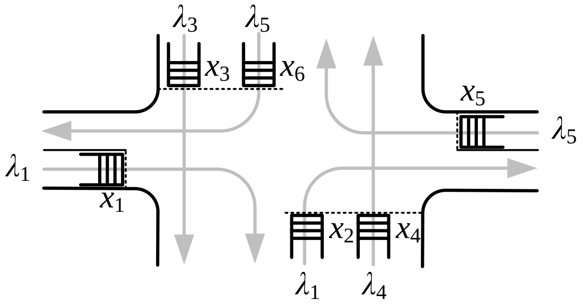


Figure 9.2: Two intersection network regarded as a single intersection.

To derive the optimal phase delay, i.e., the phase delay minimizing the sum of the average intermediate queue contents, we first focus on a two-queue network with transportation times in Section 9.1.1. For this network, given the service periods and transportation time, the optimal phase delay is determined. It is assumed that each queue only receives a single service period during a cycle and that the maximal service rates of queues on a route are identical. For the system with these assumptions, already 18 different situations exist. The results are used in Section 9.1.2

to derive the optimal phase delay for a four-queue network where fluid flows in both directions between two servers. Finally, the optimal phase delay for the traffic intersection network are derived.

9.1.1 Two-queue network

The smallest network for which fluid is transported between servers is the two-queue network depicted in Figure 9.3. Here, both servers serve a single queue and fluid is transported, with transportation time γ, to queue 2 after service by server 1. The servers have identical cycle times T, a predefined service period $(\tau_1, \tau_2 < T)$ and both queues are served once in a cycle. For this system, we derive the phase delay that minimizes the total queue contents W_2. Note that the time averaged queue contents or time averaged flow time can be used similarly as performance criterion.

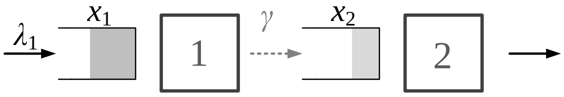


Figure 9.3: Two queue network.

The start of service of queue 2 is delayed by a phase delay ξ. The content of the intermediate queue x_2 depends on the service periods of both queues 1 and 2 and on the phase delay. Since the arrival process at queue 2 is non-constant, as it depends on the service period of queue 1 and the transportation time, the optimization problem is a timing issue, i.e., timing between arrival of fluid in queue 2 and start of service of queue 2. Given a fixed service period of queue 1 and the fixed service duration of queue 2, the total queue content during a cycle of queue 2 can be determined based on the the transportation time γ and phase-delay ξ. Denote the time that service of queue $i = 1,2$ starts at $g_i \in [0,T]$ and the service ends at $r_i \in [0,T]$. The service periods are given by

$$\begin{aligned} \tau_i &= (r_i - g_i) \bmod T, \quad i = 1,2, \\ \tau_1^\lambda &= \frac{\tau_1 - \rho_1 T}{1-\rho_1}, \\ \tau_1^\mu &= \tau_1 - \tau_1^\lambda, \end{aligned}$$

and the time that the server starts serving queue 1 at arrival rate, i.e., $x_1 = 0$, is given by $g_1^\lambda = (r_1 - \tau_1^\lambda) \bmod T$. Let a "$^-$" indicate the actual times of the arrival and the departure process of fluid at queue 2, i.e.,

$$\begin{aligned} \bar{g}_1 &= (g_1 + \gamma) \bmod T, & \bar{g}_2 &= (g_2 + \xi) \bmod T, \\ \bar{g}_1^\lambda &= (g_1^\lambda + \gamma) \bmod T, & \bar{r}_2 &= (r_2 + \xi) \bmod T, \\ \bar{r}_1 &= (r_1 + \gamma) \bmod T. \end{aligned}$$

Next, we distinguish between the start and the end of service at the different arrival rates. Service of queue 2 can start during arrival at maximal rate (indicated by phase ❶), during arrival at arrival rate (❷) or queue 2 can start service when no fluid arrives (❸). These phases are given by

$$\begin{array}{lrcr}
❶ \text{ if} & \bar{g}_1 \leq \bar{g}_2 < \bar{g}_1^{\lambda} + P_1^{\mu} T & \vee & \bar{g}_2 < \bar{g}_1^{\lambda} - (1 - P_1^{\mu})T, \\
❷ \text{ if} & \bar{g}_1^{\lambda} \leq \bar{g}_2 < \bar{r}_1 + P_1^{\lambda} T & \vee & \bar{g}_2 < \bar{r}_1 - (1 - P_1^{\lambda})T, \\
❸ \text{ if} & P_1 \bar{r}_1 < \bar{g}_2 \leq \bar{g}_1 & \vee & \bar{r}_1 + P_1 T < T,
\end{array}$$

with indicator functions

$$\mathbb{1}_1 = \begin{cases} 1 & \text{if } \bar{g}_1 > \bar{r}_1 \\ 0 & \text{otherwise} \end{cases}, \qquad \mathbb{1}_1^{\lambda} = \begin{cases} 1 & \text{if } \bar{g}_1^{\lambda} > \bar{r}_1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbb{1}_1^{\mu} = \begin{cases} 1 & \text{if } \bar{g}_1 > \bar{g}_1^{\lambda} \\ 0 & \text{otherwise} \end{cases}, \qquad \mathbb{1}_2 = \begin{cases} 1 & \text{if } \bar{g}_2 > \bar{r}_2 \\ 0 & \text{otherwise} \end{cases}.$$

Moreover, service of queue 2 can end during arrival of fluid at arrival rate (denoted by phase ①), during arrival at maximal rate (②) or in between arrivals of fluid ③. These phases are given by:

$$\begin{array}{lrcr}
① \text{ if} & \bar{g}_1 \leq \bar{r}_2 < \bar{g}_1^{\lambda} + P_1^{\mu} T & \vee & \bar{r}_2 < \bar{g}_1^{\lambda} - (1 - P_1^{\mu})T, \\
② \text{ if} & \bar{g}_1^{\lambda} \leq \bar{r}_2 < \bar{r}_1 + P_1^{\lambda} T & \vee & \bar{r}_2 < \bar{r}_1 - (1 - P_1^{\lambda})T, \\
③ \text{ if} & P_1 \bar{r}_1 < \bar{r}_2 \leq \bar{g}_1 & \vee & \bar{r}_1 + P_1 T < T.
\end{array}$$

For all combinations of phases of starting and ending service of queue 2, different queue levels arise. An example of a situation where service of queue 2 starts between arrivals (❸) and service ends while fluid arrives at arrival rate (②) is presented in Figure 9.4. The arrival rate profile is presented by the black solid lines, i.e., fluid arrives at rate μ during the interval $[\bar{g}_1, \bar{g}_1^{\lambda})$ and at rate λ during $[\bar{g}_1^{\lambda}, \bar{r}_1)$. The gray solid line indicates the service period at queue 2, i.e., queue 2 is served during $[\bar{g}_2, \bar{r}_2)$. We assume that the maximal service rates at both queues are identical ($\mu_1 = \mu_2$), to reduce the total number of situations. Note that for systems with $\mu_1 \neq \mu_2$, the mean queue contents can be derived similarly.

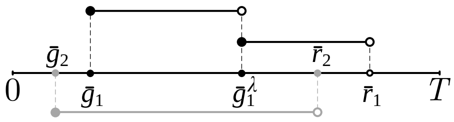


Figure 9.4: Arrival and service pattern example, corresponding to (❸,②).

The different situations based on the arrival and departure processes are indicated by Roman numbers (I-IX). A graphical representation of all different situations is presented in Figure 9.5. Note that, for ease of reading, the arrival rate profiles are identical for all situations and hence the service and arrival rates can differ between situations. Auxiliary variables, indicated by capital letters, represent the time durations at which the queue levels increase/decrease at a fixed rate.

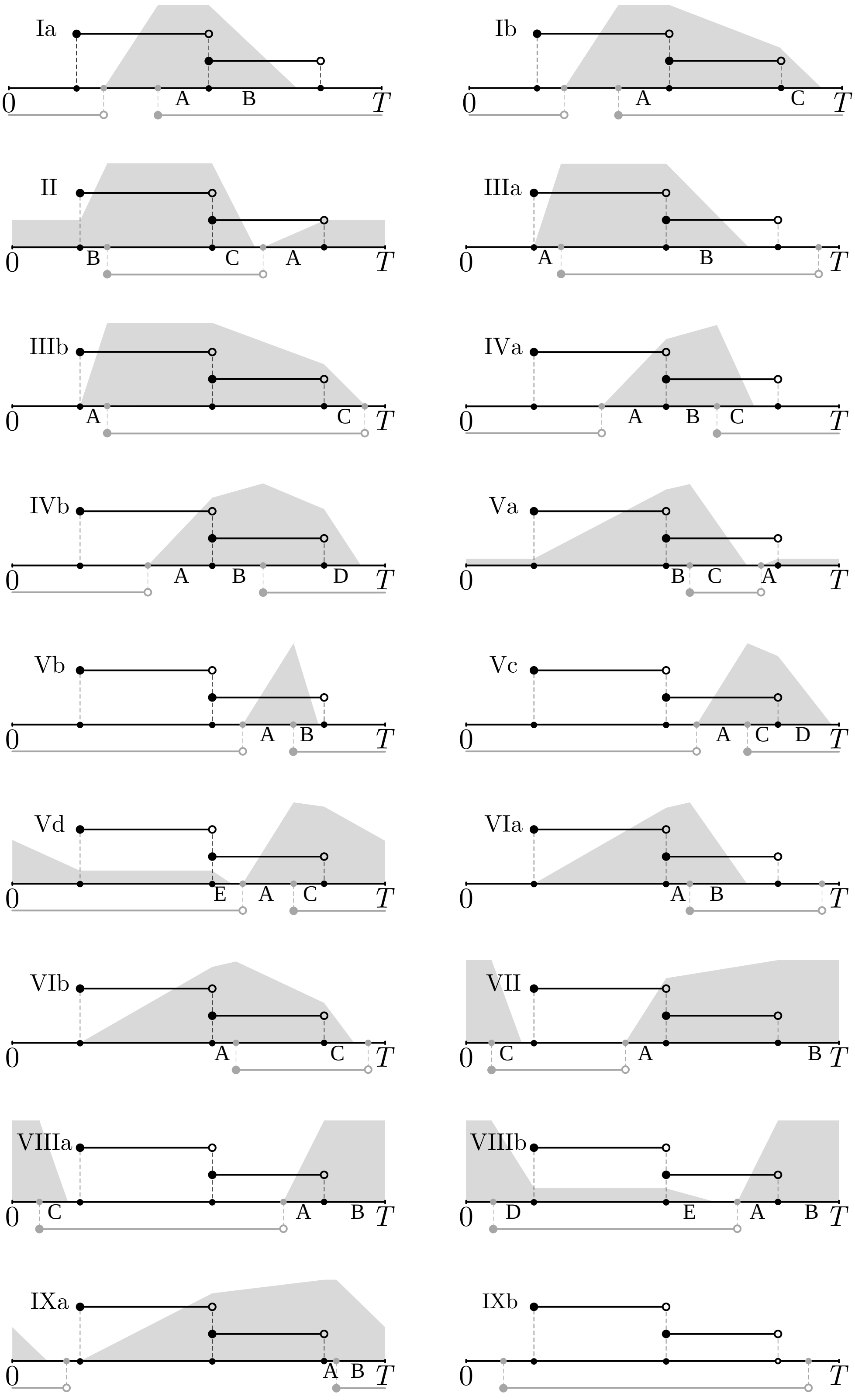


Figure 9.5: Different situations for $\mu_1 = \mu_2$. Each situation displays the arrival rate profile (black line), the total service period (gray line) and queue content (gray area) of queue 2 during a cycle.

Given the arrival process at queue 2, the problem is to find the optimal phase delay. For the entire range $\xi \in [0,T]$, the arrival and departure process of the system is represented by several situations depicted in Figure 9.5. For each of these these situations, the minimal queue contents can be easily derived. Then, the phase delay yielding the lowest queue content is the optimal phase delay. For each of the situations in Figure 9.5, the total content of queue 2 during a cycle is derived below, denoted by w_2^j for situation j. The total queue content equals the gray area in the figure. By splitting up the area into triangles, trapezoids and rectangles, the total queue lengths are computed. For stability, it is required that all fluid arriving at queue 2 can be served, i.e., $\mu_2\tau_2 \geq \lambda_1 T$.

- Situation I: (❶,①).
  Auxiliary variables

$$A = (\bar{g}_1^\lambda - \bar{g}_2) \bmod T, \qquad B = \tfrac{\mu_1(T-\tau_2)}{\mu_2-\lambda_1},$$
$$C = \tfrac{1}{\mu_2}(\mu_1(T-\tau_2) + (\lambda_1-\mu_2)\tau_1^\lambda).$$

  Total queue contents are

$$w_2^{Ia} = \frac{1}{2}\mu_1(T-\tau_2)^2 + \mu_1(T-\tau_2)\left(A+\frac{1}{2}B\right), \qquad \text{if } B \leq \tau_1^\lambda, \tag{9.1a}$$
$$w_2^{Ib} = \frac{1}{2}\mu_1(T-\tau_2)^2 + \mu_1(T-\tau_2)A + [\mu_1(T-\tau_2)+ \tag{9.1b}$$
$$+\frac{1}{2}(\lambda_1-\mu_2)\tau_1^\lambda]\tau_1^\lambda\mu_1 + \frac{\mu_2}{2}C^2, \qquad \text{if } B > \tau_1^\lambda. \tag{9.1c}$$

- Situation II: (❶,②).
  Auxiliary variables

$$A = (\bar{r}_1 - \bar{r}_2) \bmod T, \qquad B = (\bar{g}_2 - \bar{g}_1) \bmod T,$$
$$C = \tfrac{1}{\mu_2-\lambda_1}(\lambda_1 A + \mu_1 B).$$

  Total queue contents are

$$w_2^{II} = \lambda_1 A\left(\frac{1}{2}A + T - \tau_1^\lambda\right) + \mu_1 B\left(\tau_1^\mu - \frac{1}{2}B\right) + \frac{\mu_2-\lambda_1}{2}C^2. \tag{9.1d}$$

- Situation III: (❶,③).
  Auxiliary variables

$$A = (\bar{g}_2 - \bar{g}_1) \bmod T, \qquad B = \tfrac{\mu_1 A}{\mu_2-\lambda_1},$$
$$C = \tfrac{1}{\mu_2}(\mu_1 A + (\lambda_1-\mu_2)\tau_1^\lambda).$$

  Total queue contents are

$$w_2^{IIIa} = \mu_1 A\left(\tau_1^\mu - \frac{1}{2}A\right) + \frac{\mu_2-\lambda_1}{2}B^2, \qquad \text{if } B \leq \tau_1^\lambda, \tag{9.1e}$$
$$w_2^{IIIb} = \mu_1 A\left(\tau_1 - \frac{1}{2}A\right) + (\lambda_1-\mu_2)[\tau_1^\lambda]^2 + \frac{\mu_2}{2}C^2, \quad \text{if } B > \tau_1^\lambda. \tag{9.1f}$$

- Situation IV: (❷,①).
  Auxiliary variables

$$A = (\bar{g}_1^{\lambda} - \bar{r}_2) \bmod T, \qquad B = T - \tau_1 - A,$$
$$C = \tfrac{1}{\mu_2-\lambda_1}(\mu_1 A + \lambda_1 B), \qquad D = \tfrac{1}{\mu_2}(\mu_1 A + \lambda_1 B + (\lambda_1 - \mu_2)\tau_1^{\lambda}).$$

Total queue contents are

$$w_2^{IVa} = \mu_1 A\left(\frac{1}{2}A + B\right) + \frac{1}{2}\lambda_1 B^2 + \frac{\mu_2 - \lambda_1}{2}C^2, \qquad \text{if } C \le \tau_1^{\lambda}, \tag{9.1g}$$

$$\begin{aligned} w_2^{IVb} = {} & \mu_1 A\left(\frac{1}{2}A + \tau_1^{\lambda}\right) + \lambda_1 B\left(\tau_1^{\lambda} - \frac{1}{2}B\right) + \\ & + \frac{1}{2}(\lambda_1 - \mu_2)(\tau_1^{\lambda} - B)^2 + \frac{\mu_2}{2}D^2, \qquad \text{if } C > \tau_1^{\lambda}. \end{aligned} \tag{9.1h}$$

- Situation V: (❷,②).

  - Situation Va: $(\bar{r}_1 - \bar{r}_2) \bmod T \le (\bar{r}_1 - \bar{g}_2) \bmod T$
    Auxiliary variables

$$A = (\bar{r}_1 - \bar{r}_2) \bmod T, \qquad B = (\bar{g}_2 - \bar{g}_1^{\lambda}) \bmod T,$$
$$C = \tfrac{1}{\mu_2-\lambda_1}(\lambda_1 A + \mu_1 \tau_1^{\mu} + \lambda_1 B).$$

Total queue contents are

$$w_2^{Va} = \lambda_1 A\left(\frac{1}{2}A + T - \tau_1^{\lambda} + B\right) + \mu_1 \tau_1^{\mu}\left(\frac{1}{2}\tau_1^{\mu} + N\right) + \frac{1}{2}\lambda_1 B^2 \frac{\mu_2 - \lambda_1}{2}C^2. \tag{9.1i}$$

  - Situation Vb-Vd: $(\bar{r}_1 - \bar{r}_2) \bmod T > (\bar{r}_1 - \bar{g}_2) \bmod T$
    Auxiliary variables

$$A = T - \tau_2, \qquad B = \tfrac{1}{\mu_2-\lambda_1}(\lambda_1 A),$$
$$C = (\bar{r}_1 - \bar{g}_2) \bmod T, \qquad D = \tfrac{1}{\mu_2}(\lambda_1 A + (\lambda_1 - \mu_2)C),$$
$$E = \tfrac{1}{\mu_2-\lambda_1}(\lambda_1 A + (\lambda 1 - \mu_2)C - \mu_2(T - \tau_1)) \quad .$$

Total queue contents are

$$w_2^{Vb} = \frac{1}{2}\lambda_1 A^2 + \frac{\mu_2 - \lambda_1}{2}B^2, \qquad \text{if } B \le C, \tag{9.1j}$$

$$w_2^{Vc} = \lambda_1 A\left(\frac{1}{2}A + C\right) + (\lambda_1 - \mu_2)C + \frac{\mu_2}{2}D^2, \qquad \text{if } B > C \wedge D \le T - \tau_1, \tag{9.1k}$$

$$
\begin{aligned}
w_2^{Vd} = \lambda_1 A \left(\frac{1}{2} A + C + T - \tau_1^\lambda \right) + \\
&+ (\lambda_1 - \mu_2) C \left[\frac{1}{2} C + T - \tau_1^\lambda \right] - \\
&- \mu_2 (T - \tau_1) \left[\frac{1}{2} T - \frac{1}{2} \tau_1 + \tau_1^\mu \right] + \frac{\mu_2 - \lambda_1}{2} E^2, \quad \text{if } B > C \wedge D > T - \tau_1.
\end{aligned}
\tag{9.1l}
$$

- Situation VI: (❷, ③).
  Auxiliary variables

$$
\begin{aligned}
&A = (\bar{g}_2 - \bar{g}_1^\lambda) \bmod T, && B = \tfrac{1}{\mu_2 - \lambda_1} (\mu_1 \tau_1^\mu + \lambda_1 A), \\
&C = \tfrac{1}{\mu_2} (\mu_1 \tau_1^\mu + \lambda_1 A + (\mu_2 - \lambda_1)(\tau_1^\lambda - A)).
\end{aligned}
$$

  Total queue contents are

$$
w_2^{VIa} = \mu_1 \tau_1^\mu \left(\frac{1}{2} \tau_1^\mu + A \right) + \frac{1}{2} \lambda_1 A^2 + \frac{\mu_2 - \lambda_1}{2} B^2, \quad \text{if } B \le \tau_1^\lambda - A, \tag{9.1m}
$$

$$
w_2^{VIb} = \mu_1 \tau_1^\mu \left(\frac{1}{2} \tau_1^\mu + \tau_1^\lambda \right) + \lambda_1 A \left(\tau_1^\lambda - \frac{1}{2} A \right) + \tag{9.1n}
$$

$$
+ \frac{1}{2} (\lambda_1 - \mu_2)(\tau_1^\lambda - A)^2 + \frac{\mu_2}{2} C^2, \qquad \text{if } B > \tau_1^\lambda - A. \tag{9.1o}
$$

- Situation VII: (❸, ①).
  Auxiliary variables

$$
\begin{aligned}
&A = (\bar{g}_1^\lambda - \bar{r}_2) \bmod T, && (\bar{g}_2 - \bar{r}_1) \bmod T, \\
&C = \tfrac{1}{\mu_2} (\mu_1 A + \lambda_1 \tau_1^\lambda).
\end{aligned}
$$

  Total queue contents are

$$
w_2^{VII} = \mu_1 A \left(\frac{1}{2} A + \tau_1^\lambda + B \right) + \lambda_1 \tau_1^\lambda \left(\frac{1}{2} \tau_1^\lambda + B \right) + \frac{\mu_2}{2} C^2. \tag{9.1p}
$$

- Situation VIII: (❸, ②).
  Auxiliary variables

$$
\begin{aligned}
&A = (\bar{r}_1 - \bar{r}_2) \bmod T, && B = (\bar{g}_2 - \bar{r}_1) \bmod T, \\
&C = \tfrac{1}{\mu_2} \lambda_1 A, && D = T - \tau_1 - B, \\
&E = \tfrac{1}{\mu_2 - \lambda_1} (\lambda_1 A + (\lambda 1 - \mu_2) D) && .
\end{aligned}
$$

  Total queue contents are

$$
w_2^{VIIIa} = \lambda_1 A \left(\frac{1}{2} A + B \right) + \frac{\mu_2}{2} C^2, \qquad \text{if } C \le D, \tag{9.1q}
$$

$$w_2^{VIIIb} = \lambda_1 A \left(\frac{1}{2}A + B + D + \tau_1^{\mu}\right) + (\lambda_1 - \mu_2) D \left(\frac{1}{2}D + \tau_1^{\mu}\right) + \tag{9.1r}$$

$$+ \quad \frac{\mu_2 - \lambda_1}{2} E^2, \qquad \text{if } C > D. \tag{9.1s}$$

- Situation IX: (❸, ②).
  Auxiliary variables

  $$A = (\bar{g}_2 - \bar{r}_1) \bmod T, \qquad B = \tfrac{1}{\mu_2}(\lambda_1 T).$$

  Total queue contents are

  $$w_2^{IXa} = \mu_1 \tau_1^{\mu} \left(\frac{1}{2}\tau_1^{\mu} + \tau_1^{\lambda} + A\right) + \lambda_1 \tau_1^{\lambda} \left(\frac{1}{2}\tau_1^{\lambda} + A\right) + \frac{\mu_2}{2} B^2, \tag{9.1t}$$

  $$w_2^{IXb} = 0, \qquad \text{if } (\bar{g}_1 - \bar{r}_2) \bmod T > (\bar{g}_1 - \bar{g}_2) \bmod T. \tag{9.1u}$$

Optimal phase delay

For a system with known transportation time and service periods, it is desired that the phase delay is chosen such that the total queue contents are minimal. Hence, we want to minimize the total contents of queue 2, as the total contents of queue 1 are not affected by the phase delay. For a phase delay $\xi \in [0, T]$, all different situations are optimized within their range. For each situation, the total queue content W_2 is either a linear or quadratic function with respect to the phase delay. Therefore, the optimum can be quickly derived. Then, the phase delay that yields the lowest total queue contents is the optimal phase delay ξ^*.

As illustration, consider the system with $T = 36$, $\lambda_1 = 6$, $\mu_1 = \mu_2 = 12$, $g_1 = 4$, $r_1 = 18$, $g_2 = 2$ and $r_2 = 22$. Since $\rho_1 = \frac{T}{\tau_1} = \frac{1}{2}$, queue 1 is only served at maximal rate. Furthermore, a transportation time of $\gamma = 8$ is considered. For the phase delay $\xi \in [0, 36]$, the total queue contents of queue 2 are presented in Figure 9.6. In Table 9.1 the different situations and corresponding range of phase delays are presented. For each situation, the total queue contents w_2 are linear with respect to the phase delay. Minimizing w_2 for each situation yields that the phase delay $\xi \in [8 - 10]$ is optimal.

It is shown that the optimal phase delay for the two-queue network can be derived by finding the minimal queue content for all possible situations over the range $\xi \in [0, T]$. Next, a similar approach is presented for a network is where fluid flows in both directions between servers.

9.1.2 Four-queue network

Here, the queue contents of a network with two servers and four queues are discussed. In the considered network, presented in Figure 9.7, fluid flows to and from

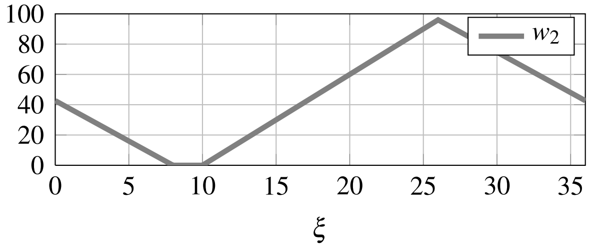


Figure 9.6: Total queue contents w_2 for $\xi \in [0, 36]$.

range ξ	situation	w_2
$[0-8]$	VII	$5\frac{1}{3}(8-\xi)$
$[8-10]$	IXb	0
$[10-26]$	IIIb	$6(\xi-10)$
$[26-28]$	Ib	$96-5\frac{1}{3}(\xi-26)$
$[28-36]$	VII	$85\frac{1}{3}-5\frac{1}{3}(\xi-28)$

Table 9.1: Situations of queue 2.

each server via two different routes. Transportation from server 1 to server 2 requires a transportation time γ_1 and a transportation time γ_2 is required for the opposite direction. Similar to the two queue network, we derive the optimal phase delay minimizing the total queue contents of the intermediate queues $(w_2 + w_4)$.

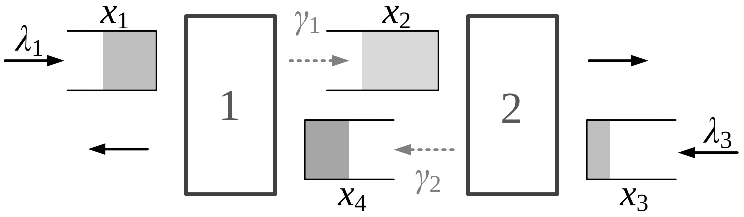


Figure 9.7: System with delay.

Given the service periods of each queue in the network, the intermediate queue contents (x_2 and x_4) can be derived based on the transportation times and phase delay ξ similar as for the two queue network presented in Figure 9.3. Without loss of generality, we assume that the service periods at server 1 are fixed. Hence, the service times at server 2 have a phase delay ξ. Then, the total contents of queue 2 during a cycle can be derived by (9.1). The contents of queue 4 can also be derived by (9.1), given that

$$\bar{g}_1 = (g_3 + \gamma_2 + \xi) \bmod T, \qquad \bar{g}_2 = g_4,$$
$$\bar{g}_1^{\lambda} = (g_3^{\lambda} + \gamma_2 + \xi) \bmod T, \qquad \bar{r}_2 = r_4,$$
$$\bar{r}_1 = (r_3 + \gamma_2 + \xi) \bmod T.$$

Optimal phase delay

The optimal phase delay for the four-queue network is derived similar to deriving the optimal phase delay for the two-queue system, discussed in Section 9.1.1. For a phase delay $\xi \in [0,T]$, all different situations are derived. Next, the phase delay for which the sum of the average queue contents is minimal, results in the optimal phase delay.

For example, consider the network presented in Section 9.1.1 again. This network is extended with queues 3 and 4 with $\lambda_3 = 4$, $\mu_3 = \mu_4 = 12$, $g_3 = 4$, $r_3 = 26$, $g_4 = 2$, $r_4 = 20$ and $\gamma_1 = \gamma_2 = 8$. Since $\rho_3 < \frac{T}{\tau_3}$, queue 3 is served at both maximal and arrival rate during a cycle. For the phase delay $\xi \in [0,36]$, the total queue contents w_2 and w_4 are presented in Figure 9.8. For this example, all situations and corresponding phase delay ranges of queue 4 are also in Table 9.2. For $\xi^* = 10$, the sum of the total intermediate queue contents is minimal ($55\frac{2}{3}$).

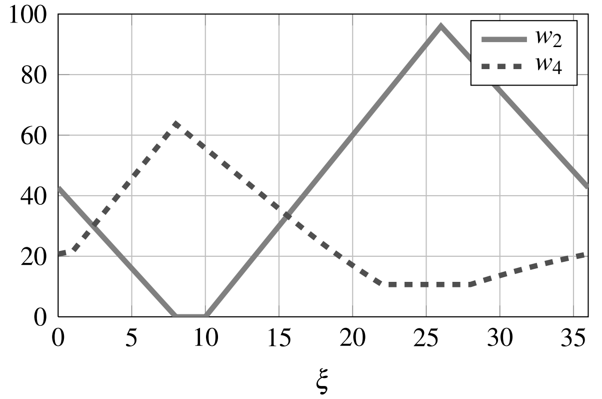


Figure 9.8: Total queue contents w_2 and w_4 for $\xi \in [0,36]$.

range ξ	situation	w_4
$[1-4]$	VII	$21\frac{2}{3}+6(\xi-1)$
$[4-8]$	IVb	$45\frac{2}{3}+6(\xi-4)$
$[8-16]$	VIb	$63\frac{2}{3}-4(\xi-8)$
$[16-19]$	VIa	$\frac{1}{12}\xi^2-\frac{20}{3}\xi+117$
$[19-22]$	IIIa	$\frac{1}{12}\xi^2-\frac{20}{3}\xi+117$
$[22-26]$	II	$10\frac{2}{3}$
$[26-28]$	VIIIa	$10\frac{2}{3}$
$[28-1]$	VIIIb	$10\frac{2}{3}+\frac{14}{9}(\xi-28) \bmod T-$ $-5\frac{1}{27}((\xi-28) \bmod T)^2$

Table 9.2: Situations of queue 4.

Traffic Example

We now consider the traffic network depicted in Figure 9.1 again with parameters

$$\Sigma = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \end{bmatrix}, \quad \lambda = \begin{bmatrix} 6 \\ 0 \\ 4 \\ 3 \\ 4 \\ 0 \end{bmatrix}, \quad \mu = \begin{bmatrix} 12 \\ 12 \\ 12 \\ 12 \\ 12 \\ 12 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{9.2}$$

For the abstracted system, depicted in Figure 9.2, the optimal service schedule has a cycle time $T = 36$, service periods $\tau = \begin{bmatrix} 18 & 20 & 12 & 10 & 22 & 18 \end{bmatrix}^\top$ and $g = \begin{bmatrix} 4 & 2 & 24 & 26 & 2 & 6 \end{bmatrix}^\top$.

As the service rates of the intermediate queues are equal to their preceding queues, the optimal phase delay can be derived in a similar way as presented in Section 9.1.2. The resulting minimal average weighted queue contents J_W, for a time delay $\gamma_1 = \gamma_2 = \gamma \in [0, T]$ is presented in Figure 9.9. The upper figure presents the total queue contents and the lower figure presents the corresponding phase delay. For $14 < \gamma < 18$ and $32 < \gamma < 36$ a range of phase delays results in the minimal average queue contents.

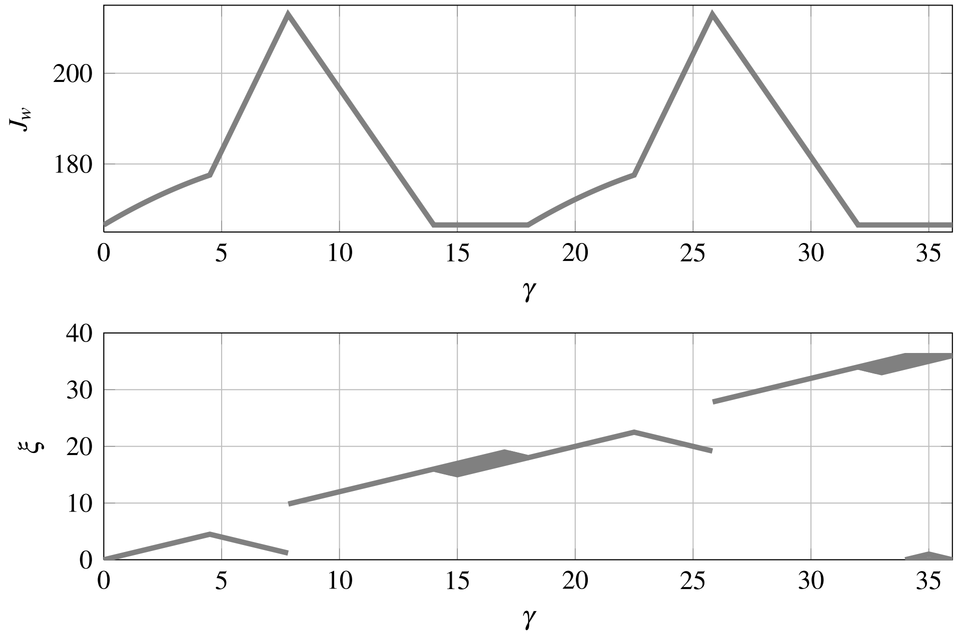


Figure 9.9: Minimal total queue content and corresponding phase delay.

Note that the resulting service schedule is not necessarily the optimal service schedule for the network of intersections with transportation times. However, as the queue contents of the non-intermediate queues have been optimized and an optimal phase delay has been derived to minimize the intermediate queue contents, we believe that this approach results in a good schedule. For networks with multiple intermediate queues the intermediate queue contents can be similarly derived. For networks with $S > 1$ (multiple) servers, $S-1$ phase delays have to be optimized. This can be done with a similar approach and is a tedious job, due to a large number of different situations for the arrival and service processes.

9.2 Stochastic arrival process

Throughout the thesis, deterministic fluid flow switching servers are considered. In reality, inter-arrival times of products at manufacturing systems or vehicles at traffic intersections can fluctuate over time. In this section, the effect of stochastic arrivals are discussed and a method to derive service schedules is presented. A model that often gives a (good) description of fluctuations in the arrival process is the Poisson process where the time between consecutive arrivals is *exponentially* distributed and *independent* of other inter-arrival times.

We consider a system with discrete objects arriving according to a Poisson arrival process. Still, the service times are assumed to be constant, which seems reasonable for, e.g., vehicles leaving an intersection. We present approximations for the average queue length and average waiting time in systems with fixed time control and arrivals described by a Poisson process. Using these approximations, service schedules are derived which are compared to the service schedules derived from deterministic fluid flow switching servers.

The outline of this section is as follows: In Section 9.2.1, a short system description is given. Section 9.2.2 provides a brief comparison between discrete deterministic and stochastic simulations. Section 9.2.3 presents a brief overview of existing approximations of the average queue length and average waiting time. An approximation of the average queue length and average waiting time for a system where the queue is served multiple times is presented in Section 9.2.4.

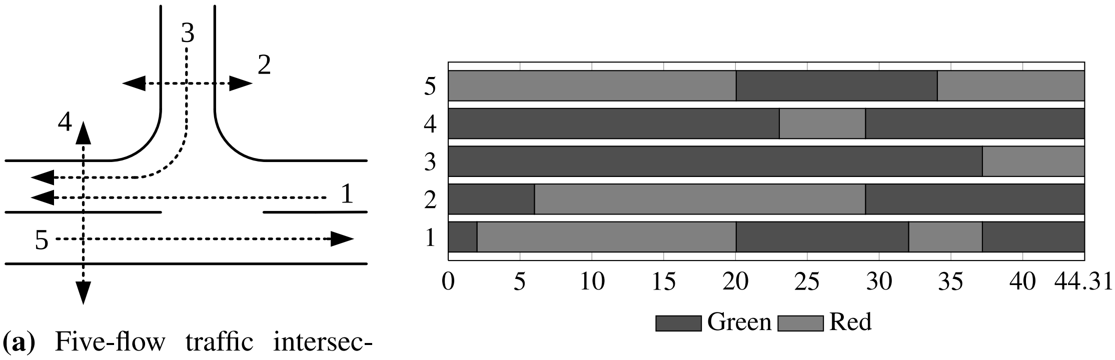
9.2.1 System description

The considered system is a discrete event system, i.e., with discrete objects, e.g., products or vehicles. Objects arrive at queue n according to a Poisson process with arrival rate λ_n, which is assumed to be constant. The Poisson process is characterized as a renewal process and inter-arrival times t_A are exponentially distributed, i.e., $P(t_A \leq t) = 1 - e^{-\lambda_n t}$ for all $t \geq 0$. The duration of service period τ_n and cycle time T are fixed. The service time of objects of queue n, during the service period, is constant and equal to $\frac{1}{\mu_n}$. The load of the system is denoted by $\rho_n = \frac{\lambda_n}{\mu_n}$ and the degree of saturation, i.e., the average capacity used by the server, is denoted by $\rho_n^* = \frac{\lambda_n T}{\mu_n \tau_n}$.

When the service period of queue n ends, i.e., at the start of an idle period, and when service is not completely finished, we assume that the service of the object is preempted and the service resumes at the beginning of the next service period at the point where it was interrupted (preemptive resume). The queue length increases by one when an object arrives and the queue length decreases by one when an object has been served (entirely).

9.2.2 Discrete-event simulations

In this section, we investigate the applicability of the derived service schedules for discrete-event systems. Recall the five-flow traffic intersection introduced in Chapter 4, presented in Figure 9.10a.



(a) Five-flow traffic intersection.

(b) Optimal fluid flow service schedule.

Figure 9.10: Layout (left) and graph (right) of an intersection with three vehicle lanes (1,3 and 5) and two pedestrian lanes (2 and 4).

For this system, the parameters are

$$\Sigma = \begin{bmatrix} 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 2 & 6 & 0 & 5 & 0 \\ 3 & 0 & 3 & 0 & 5 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \lambda = \begin{bmatrix} 3 \\ 0.1 \\ 1 \\ 0.1 \\ 4 \end{bmatrix}, \quad \mu = \begin{bmatrix} 10 \\ 2 \\ 10 \\ 2 \\ 10 \end{bmatrix},$$

the weight factors for all queues are equal to 1 and restrictions on the cycle time, minimal green periods and maximal queue contents are given by

$$\begin{aligned} 40 \leq T \leq 60&, \\ \tau^{\min} = \begin{bmatrix} 3 & 6 & 3 & 6 & 3 \end{bmatrix}&, \\ x_n^{\max} = 100, \qquad n &= 1,2,\ldots,N. \end{aligned}$$

The optimal periodic behavior for this system has been derived in Chapter 4. If Θ = 2 and $G = 4$, i.e., the maximal number of service periods for a queue in a cycle is two and at most four groups are allowed in a sequence, the service schedule resulting in optimal behavior $J_w^* = 45.35$ is presented in Figure 9.10b. The average queue contents W_n of each queue and the total average queue content of the system J_w are presented in Table 9.3. In this table a comparison is shown between the performance of the fluid model and for the discrete event model with deterministic and Poisson arrivals. The latter results are generated using 400 simulations where in each simulation 30000 objects have been served. The average queue contents are presented together with the 95% confidence intervals.

	Fluid	Discrete	Discrete
		Deterministic	Poisson
W_1	10.986	11.432	12.437 ± 0.387
W_2	0.537	0.600	0.617 ± 0.064
W_3	17.341	17.475	17.544 ± 0.475
W_4	1.743	1.811	1.837 ± 0.137
W_5	14.745	15.196	15.511 ± 0.404
J_w	45.352	46.514	47.945 ± 0.364

Table 9.3: Average queue contents for the schedule depicted in Figure 9.10b.

Note that both simulations with discrete objects result in slightly larger queue contents, as expected. This service schedule is able to serve all arrivals for the system with Poisson arrivals, since additional capacity is available at each queue.

Remark 9.2.1. *In this example, the service schedule is able to serve all objects, regardless of a deterministic or stochastic arrival process, since additional service capacity is available at each queue. This is not always the case, as servers for the derived service schedules, based on fluid flow models, often switch to serve another queue directly after the queue has been emptied. For these queues, the service capacity is fully utilized and variability in the arrival process causes the waiting time (and thus also queue content) to go to infinity, resulting in undesired behavior.*

We offer several solutions to prevent this undesired behavior discussed in the remark above.

- Take possible fluctuations into account in the deterministic fluid flow switching server schedule derivation process. The schedules can be derived such that additional service capacity is present at all queues. This is possible by deriving schedules for the system by considering larger arrival rates or imposing a maximal capacity threshold, for instance 90%. Both can be easily implemented, the latter, e.g., by adapting (4.15).

- Take the stochastic arrival process into account and define the problem as a *non-linear programming problem* (NLP). Here, the average waiting time is approximated and used as objective function for the NLP problem. The method of deriving the schedule and constraints do not change. Several approximations are discussed below.

9.2.3 Approximations for average delay

Approximation formulas for the average delay $E[\varphi_n]$ (or waiting time) of an object in queue n are presented below. In steady-state, the average number of waiting objects $E[x_n]$ relates to the average delay via Little's law, i.e.,

$$E[x_n] = \lambda_n E[\varphi_n]. \tag{9.3}$$

A well-known, and still widely used, approximation of the average delay of objects at queue n, developed by [96], is given by

$$E[\varphi_n] = \frac{(T-\tau_n)^2}{2T(1-\rho_n)} + \frac{\rho_n T^2}{2\tau_n(\mu_n\tau_n - \lambda_n T)} - 0.65\left(\frac{T}{\lambda_n^2}\right)^{\frac{1}{3}}\left(\frac{\lambda_n T}{\mu_n\tau_n}\right)^{2+5\frac{\tau_n}{T}}. \tag{9.4}$$

This formula is based on a $M/D/1$ system, i.e., exponentially distributed inter-arrival times and deterministic service times, with cyclic service and idle periods of the server. Each cycle consists of a single service period τ_n and a single idle period $T-\tau_n$. The first term of (9.4) corresponds to the delay of the fluid model, which is a fairly good approximation for the delay at low ρ_n^*, as the influence of the random nature of the arrivals is minor. The second term of (9.4) compensates for the random nature of the arrivals. It is the average delay for a $M/D/1$ system if the service times are stretched with a factor $\frac{T}{g}$, i.e., the server is always serving queue n and objects have a service time of $\frac{T}{\mu_n\tau_n}$. The third term of (9.4) is a correction term based on simulation results and the value of this term accounts for 5-15% of the total value.

Another approximation for the considered system is given in [22]:

$$E[\varphi_n] = \frac{l_n}{\lambda_n} + \frac{(T-\tau_n)^2}{2T(1-\rho_n)} + (\rho_n^*)^4\frac{T-\tau_n}{2(1-\rho_n)(\mu_n\tau_n - \lambda_n T)}, \tag{9.5}$$

with l_n being the expected number of objects in an $M/D/1$ system, i.e.,

$$l_n = \rho_n + \frac{\rho_n^2}{2(1-\rho_n)}.$$

In (9.5), the first term corresponds to the average delay of objects in an $M/D/1$ system. The second term is the average delay for the fluid model and the third term is a correction term. Via a discrete-event simulation, these terms are discussed below. The discrete-event simulation model has only three events, arrival of an object, queue length measurement at regular points in time and departure of an object. For reliability of the results, 1000 cycles are considered per simulation and

100 simulations are performed. In Figure 9.11, the mean number of objects during a single period are presented for the system with parameters $\lambda_n = 0.15$, $\mu_n = 0.5$, $\tau_n = 50$ and $T = 100$ ($\rho_n^* = 0.6$). The solid lines represent the mean number of objects for the system with Poisson arrivals and the dotted lines represent the sum of l_n and the number of objects for deterministic arrivals, i.e., the first two terms in (9.5), if translated to the mean number of objects via Little's law. The third term of (9.5) is a correction term which compensates for the underestimation, as can be seen in the figure. This term is based on an educated guess multiplied by the expected number of waiting objects at the end of a service period. As shown in [22], approximation (9.5) yields better results than (9.4).

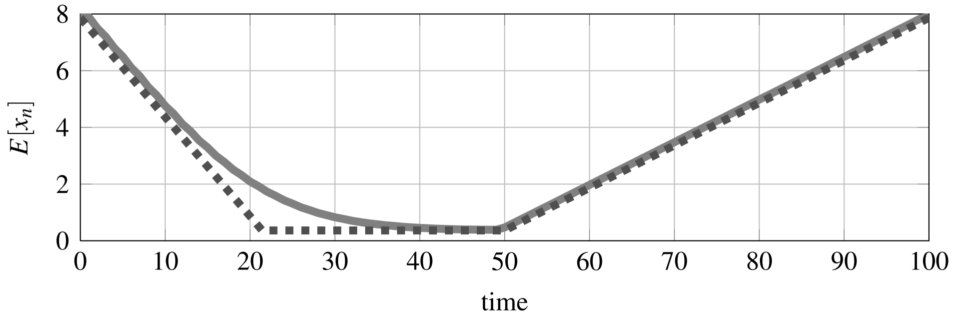


Figure 9.11: Mean number of objects during a cycle (solid) and number of objects for the fluid system plus l_n (dashed).

9.2.4 Multiple service periods

In this thesis, fluid flow switching servers are considered that can serve a queue multiple times in a cycle. An example of the average queue content during a cycle where the queue is served twice is presented in Figure 9.12 by the solid line. In each cycle, the queue is served for a duration of 10 time units ([0,10]) and 40 time units ([30,70]). The intersected line indicates the sum of the average queue content for the deterministic system added to l_n. It can be clearly seen that during the first service period in the cycle not all objects can be served that have arrived during the preceding idle period.

For the systems with Poisson arrivals, this phenomenon has been addressed in [34]. In that work, the authors assumed stability of all *subcycles*, i.e., a consecutive red and green period. In [95], approximations are presented for the average queue content for the considered system which is able to serve a queue multiple times in a cycle without a stability requirement on subcycles, identical to the problems discussed in Chapter 4. Next, these approximations are used to derive service schedules via NLP and the performance of the approximations is evaluated by comparing the performances at multiple different scenarios. The approximation, based on the approximation derived in [96], which results in schedules with the overall best per-

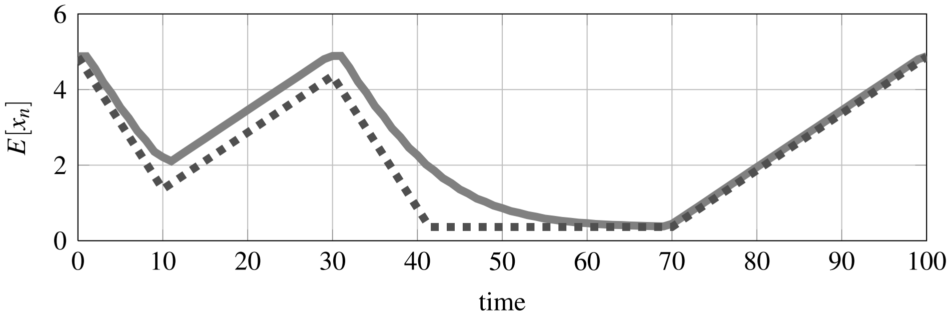


Figure 9.12: Mean number of objects during a cycle (solid) and number of objects for the fluid system plus l_n (dashed).

formance is given by

$$E[\varphi_n] = \rho_n^* + \frac{\rho_n^{*2}}{2(1-\rho_n^*)} + \varphi_n^{\text{fluid}}, \tag{9.6}$$

where φ_n^{fluid} is the average waiting time for the fluid flow model, see Chapter 4 for more details. Furthermore, [95] also implemented approximation (9.6) in a tool to derive service schedules.

9.3 Supply networks

A supply network is a generalization of the original concept of a supply chain where mainly linear or tree-structured facilities and material flows are considered. In today's network economy more complex structures exist where, in principle, any of the involved components can supply each other, and these are captured in the term 'supply network'.

Managers face increasing pressure to control inventories and costs along the network while maximizing customer service performance. Their most important task is organizing the flow of products and information within the supply network. This includes both the design of the network as well as the control of the material flow. Due to evolving networks, it remains a major challenge to achieve optimal performance. In particular, managing uncertainties, such as the buyers demand, material availability or variabilities in transport and service times, is of great importance. This problem is classified as a Supply Chain Operations Planning problem, with the objective to coordinate the release of items in the supply network at minimal cost, cf. [61, 60]. They also derived optimization models considering either the network structure or control policies. Typically, a control strategy is given and influence of the parameters on the dynamics are studied, see for instance [31].

This section is partly based on [109].

For the problem of finding an optimal control strategy for supply networks without any strategy assumptions a priori we believe techniques from control theory are helpful. By way of illustration a supply network is considered where a retailer can order products at multiple sources with different lead times and costs, and faces an unknown customer demand. A well-known method from control theory is introduced to find optimal control strategies for flows in demand-driven supply networks, independent of structural assumptions about the network and therefore without any prior assumptions of a certain family of strategies. It is not our aim to derive new policies, as the resulting policies are well known, but our aim is to show that ideas from control theory can be applicable for supply networks and that these therefore can be a stepping stone for future results. Note that introducing control theory applications to a production-inventory problem is not a new phenomenon, cf. [76], [64] and [57] and references therein.

An additional complexity in the management of material flows through a supply chain is introduced by the organizational structure and barriers within the network. Nowadays multiple sites worldwide are working together to deliver a product, while reporting to different organizational units within the corporation. Each of these sites has its unique culture, constraints, and objectives. Therefore, for some networks complete centralized control of material flows would be optimal but may not be feasible. For these cases, global information is required throughout the network with proper communication between warehouses.

However, proper communication is not always possible, as for instance indicated in [29] and [30]. One could think of complicated communication due to the infrastructure or competition. Therefore, the combination of a network with control laws concerning multiple states of the system and poor communication could yield non-optimal results. In control theory similar problems exist where the complete state of the system is not directly measurable. A solution to this problem is to derive *observers* which estimate the state based on measurements, see for instance [75, 85]. We show that this method is also applicable in a supply network setting. Measuring inputs and outputs of the system for a while enables the observer to reconstruct unknown states. In other words, by estimating the unknown states in the supply network the control laws can be perfectly utilized. The estimation is performed using a state estimator which employs only the available directly measurable input and output signals. [71] introduced this method for linear systems. Hence, the problem of designing controllers for systems with incomplete state measurements is equivalent to constructing observer-based controllers.

The remainder of this section is organized as follows. The supply network dynamics and constraints are presented in Section 9.3.1. Section 9.3.2 introduces constrained robust optimal control. An illustrative example of a supply network is presented in Section 9.3.3, used to determine the optimal control. State dependent control based on observers is presented with an example in Section 9.3.5. Conclusions are provided in Section 9.3.7.

9.3.1 System description

The supply network consists of *production units*, introduced by [16], separated by stock points. A production unit may be a single machine, a production line or a production facility. A *supply network* is a set of production units, separated by controlled stock points, to which production orders are released. We consider the supply network as a discrete time controlled dynamical system with uncertainty. Two representations of a network consisting of two stock points x_1 and x_2 with a production unit PU in between is presented in Figure 9.13. The upper representation in OR-framework is well known. The lower figure represents the network as a discrete time controlled dynamical system. An item in stock x_2 has *lead time* L, i.e., the time between the release of an order and availability of the goods in the next stock point x_1. This lead time is assumed to be deterministic and integer, and is divided over $L-1$ intermediate stock points $x_{1,L-1},\ldots,x_{1,1}$, cf. [64].

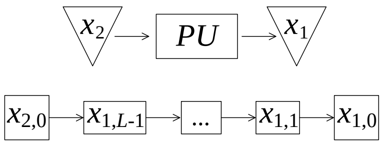


Figure 9.13: Representation of two stock points x_1 and x_2 and production unit PU with lead time L in OR-framework (top) and as a discrete time dynamical system (bottom).

The dynamics of a supply network with uncertain demand can be written as

$$x(k+1) = Ax(k) + Bu(k) + Ed(k), \tag{9.7}$$

where $x(k)$ denotes the state vector. For a supply network one can think of states describing the physical inventory of an item at a warehouse, backlog of an item or the number of ordered items waiting to be received. The vector $u(k)$ denotes the input, which consists of order quantities of items at warehouses. This vector belongs to the set $\mathscr{U}(k) \subseteq \mathbb{R}^{n_u}$. The exogenous disturbance $d(k) \in \mathscr{D}(k) \subseteq \mathbb{R}^{n_d}$ describes the customer demand and is assumed to be restricted to $[0, d^{max}]$. Linear combinations between these vectors are implied by system matrix $A \in \mathbb{R}^{n_x \times n_x}$, input matrix $B \in \mathbb{R}^{n_x \times n_u}$ and disturbance matrix $E \in \mathbb{R}^{n_x \times n_d}$.

This system is constrained by

$$Fx(k) + Gu(k) \leq g,$$

with $F \in \mathbb{R}^{n_g \times n_x}$, $G \in \mathbb{R}^{n_g \times n_u}$, $g \in \mathbb{R}^{n_g}$ and $n_g \in \mathbb{N}_0$. These constraints are used to model lower bounds, e.g. nonnegativity constraints of the physical inventories or order quantities, or upper bounds, e.g limited storage capacity or maximum order quantity. By setting nonnegative inventories the allowed order quantities become state-dependent, e.g. a warehouse can not send more items than available.
We consider the problem of finding optimal inputs $u(k)$ to the system with respect

to the constraints and costs on state and control variables.
The costs $C(k)$ are described by

$$C(k) = Qx(k) + Ru(k),$$

where one can denote costs on inventory, backlog etc. in state cost matrix $Q \in \mathbb{R}^{1\times n_x}$ and cost depending on the input, e.g. order costs, in input cost matrix $R \in \mathbb{R}^{1\times n_u}$.

9.3.2 Constrained robust optimal control

Any policy (sequence of control input decisions) produces a sequence of costs. The goal in constrained robust optimal control is to find inputs that guarantee satisfaction of the constraints for all possible combinations of disturbances and that are favorable with respect to the resulting cost distribution due to the disturbances. In many practical situations, a stochastic description of the uncertainty may not be available, and one may have information with less detailed structure, such as bounds on the magnitude of the uncertain quantities. Under these circumstances one may use a *min-max* approach, whereby the worst possible values of the uncertain disturbance within the given set are assumed to occur, see for instance [17]. For this system, the approach is to minimize the worst case cost via a minimum over u and maximum over d. Given that system (9.7) is in state $x(t_0)$ at time $t_0 = 0$ and given a horizon $K \in \mathbb{N}$, we are looking for an optimal control input sequence $(u^*(k))_{k=0}^{K-1}$, i.e., sequence of all $u^*(k)$ from $k = 0$ until $k = K-1$, such that the cost-to-go is defined by

$$J^*[k](x(k)) = \min_{u(k)} J[k](x(k), u(k)) \tag{9.8}$$

subject to

$$\begin{aligned} Fx(k) + Gu(k) &\le g, \\ Ax(k) + Bu(k) + Ed(k) &\in \mathscr{X}^{(k+1)}, \\ \forall d(k) &\in [0, d^{max}]. \end{aligned}$$

$$J[k](x(k), u(k)) = Qx(k) + Ru(k) + \max_{d(k)\in[0,d^{max}]} J^*[k+1](Ax(k) + Bu(k) + Ed(k))$$

for $k = K-1, \ldots, 0$. Here $x(k)$ is the state vector at time $t_0 + k$ given the system started in $x_0 = x(t_0)$ and was exposed to input sequence $(u(j))_{j=0}^{k}$ and disturbance sequence $(d(j))_{j=0}^{k}$ and where the system remained in the set of *feasible* states $\mathscr{X}^{(k)}$. This set is described by

$$\mathscr{X}^{(k)} = \{x \in \mathbb{R}^{n_x} | \forall d \in \mathscr{D} \exists u \in \mathbb{R}^{n_u} \text{ with}$$

$$Fx^{(k)} + Gu^{(k)} \leq g \wedge Ax^{(k)} + Bu^{(k)} + Ed^{(k)} \in \mathscr{X}^{(k+1)}\}$$

meaning that for all possible disturbances there exists an input which respects the constraints and makes sure that the state at the next step is within the feasible set. As boundary condition zero costs are assumed

$$J^*[K](x(K)) = 0$$

and

$$\mathscr{X}^{(k)} = \{x \in \mathbb{R}^{n_x} | Fx \leq g\}.$$

Then, the optimal control law u^* can be computed by dynamic programming (DP) over k

$$\begin{aligned} J[K](x(K)) =& 0, \\ J[k](x(k)) =& \min_{u(k)\in\mathscr{U}} Qx(k) + Ru(k) + \\ &+ \max_{d(k)\in[0,d^{max}]} J[k+1](Ax(k) + Bu(k) + Ed(k)), \qquad k = 0, 1, ..., K-1. \end{aligned} \tag{9.9}$$

Applying this algorithm results in the optimal cost-to-go $J[0](x(0))$. Also, if $u^*(k)$ minimizes the right hand side of (9.9) for both $x(k)$ and k, the policy $(u^*(k))_{k=0}^{K-1}$ is optimal. In each iteration the DP algorithm gives the optimal cost-to-go for every possible state, denoted by $J^*[k]$.

Using this method, optimal control of a supply network is derived by multiparametric linear programming (mpLP). In the next section this method is presented by using an illustrative example of a supply network.

9.3.3 Illustrative example

A supply network is considered to illustrate the method of finding optimal inputs with robust optimal control as described in the previous section. A graphical representation of this network is given in Figure 9.14, where nodes are stages in the network, solid arcs denote that an upstream stage supplies a downstream stage and intersected arcs denote the upstream orders. The network consists of a retailer R, the manufacturer M as first source and the subcontractor S as the second source. Note that this is not a convergent system, i.e., there is no assembly of items. For each step k, the retailer faces an uncertain demand $d(k)$ by its customers and can choose to order $u_1(k)$ number of items with lead time 2 from the manufacturer and $u_2(k)$ number of items with lead time 1 from the subcontractor. The inventory level of the retailer is denoted by x_R. The intermediate states x_{R1} and x_{R2} denote the number of intermediate items, i.e., $x_{R2}(k)$ are the number of items ordered at M at time $k-1$

and $x_{R1}(k)$ are the number of items ordered at M at time $k-2$ plus the number of items ordered at S at time $k-1$. The number of items supplied to the customer is denoted by $s(k)$. Furthermore, the retailer supplies items to the customer with a lead time of 1, and $x_d(k)$ denotes the number of ordered items at time $k-1$.

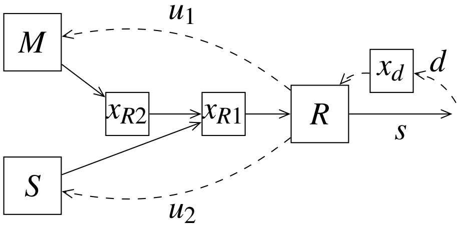


Figure 9.14: Supply network consisting of a retailer (R) and two suppliers (M and S). The retailer faces demand d, orders u_1 and u_2, and sells s items.

The dynamics of the system are described by

$$x(k+1) = \begin{bmatrix} 1 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} d(k),$$

with $x = \begin{bmatrix} x_R & x_{R1} & x_{R2} & x_d \end{bmatrix}^\top \in \mathbb{R}^4_+$ with $\mathbb{R}_+ = [0,\infty)$ and $u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top \in \mathbb{R}^2_+$. The orders $u_j(k)$ (j=1,2) are bounded by $[0, u_j^{max}]$, representing for instance a maximal transportation capacity. Demand $d(k)$ is bounded by $[0, d^{max}]$. We assume that the retailer always meets the customers demand, i.e., $s = x_d$. Therefore, all buffer levels are nonnegative, i.e., no backlog. This introduces an extra constraint on the system, $d^{max} \leq u_1^{max} + u_2^{max}$ since the demand can always be fulfilled by both suppliers. Furthermore, it is assumed that the manufacturer and subcontractor to have an infinite stock, i.e., retailer orders can always be supplied. The retailer faces c^+ inventory costs per item on inventory in stock (x_R) and order costs c_{u_1} and c_{u_2} per ordered item for ordering at the manufacturer and the subcontractor, respectively. Total costs are expressed by

$$C(k) = c^+ x_R(k) + c_{u_1} u_1(k) + c_{u_2} u_2(k). \tag{9.10}$$

Throughout the section the following parameters are assumed: $d^{max} = 8$, $u_1^{max} = 9$, $u_2^{max} = 5$, $c^+ = 1$, $c_{u_1} = 1$ and $c_{u_2} = 4$.
By solving optimization problem (9.8), it follows that the feasible region, i.e., set of feasible states $\mathscr{X}^\infty$, is given by

$$x_R + x_{R1} - x_d \geq 0, \tag{9.11}$$

$$x_R + x_{R1} + x_{R2} - x_d \geq 3, \tag{9.12}$$

where (9.11) makes sure that the physical inventory of x_R is nonnegative at a single order with maximal demand and (9.12) makes sure that the physical inventory of x_R remains nonnegative at a sequence of orders with maximal demand. The resulting robust optimal control is given by

$$\begin{aligned} u_1^*(k) &= \min[\max(14 - z(k), 0), 6], && (9.13) \\ u_2^*(k) &= \max(8 - z(k), 0), && (9.14) \end{aligned}$$

with *echelon inventory position* $z(k) = x_R(k) + x_{R1}(k) + x_{R2}(k) - x_d(k)$. In this policy, items are ordered from the manufacturer M with a maximal amount of 6 items when the inventory falls below threshold $14 - z(k)$, see (9.13). Items are ordered from the subcontractor S when the inventory falls below $8 - z(k)$, see (9.14). In the next subsection these order-up-to levels are determined analytically. Note that this is not a new policy, as [92] proved that this dual-basestock policy is optimal for this system. However, the robust optimal control approach leads to the optimal ordering policy without any prior assumption about the structure of the policy. This indicates that the control theory approach is promising.

9.3.4 Order-up-to level

In order to process all orders without backlog we have $u_1^{max} + u_2^{max} \geq d^{max}$ and $u_1 + u_2 \leq d^{max}$. Let us denote the desired order quantity at manufacturer M by $\bar{u}_1$ and the maximal remaining order quantity at subcontractor S by $\bar{u}_2 = d^{max} - \bar{u}_1$. Given that backlog is not allowed, the lower bound on the order-up-to level of the retailer is given by

$$L = 2d^{max} - \bar{u}_2 = d^{max} + \bar{u}_1. \qquad (9.15)$$

Note that this lower bound is also the desired level since a higher order-up-to level increases the inventory costs. The order policy is given by

$$\begin{aligned} u_1(k) &= \min[\max(L - z(k), 0), \bar{u}_1)], && (9.16a) \\ u_2(k) &= \min[\max(L - \bar{u}_1 - z(k), 0), \bar{u}_2], && (9.16b) \end{aligned}$$

where the retailer orders between zero and $\bar{u}_1$ products at M and the remainder, if necessary, at S. If $c_{u_1} < c_{u_2}$, one is inclined to order the products at the cheaper manufacturer M. However, due to the larger lead time, ordering more products at M results in a larger order-up-to level, see (9.15), and complementary higher inventory costs. Therefore, a trade-off exists between order costs and inventory costs, which are related to $\bar{u}_1$. The desired order quantities $\bar{u}_1$ can be derived by regarding the extreme cases: maximal order costs C_O and maximal inventory costs C_I. With robust control the maximal costs ($\max(C_O, C_I)$) are minimized. The maximal cost of ordering is given by

$$C_O = (c_{u_1} - c_{u_2})\bar{u}_1 + c_{u_2} d^{max}, \tag{9.17}$$

which can be derived by regarding ordering at maximal demand. Note that in this case, the inventory costs are zero. The maximal costs spent on inventory, denoted by C_I, depend on the maximal inventory level. The inventory level is maximal if $d = 0$ for at least two time-units and is equal to the order-up-to level L. Therefore, maximal costs spent on inventory are:

$$C_I = c^+ L. \tag{9.18}$$

It can be seen that C_O is decreasing and C_I is increasing for increasing $\bar{u}_1$. Therefore, maximal costs are minimal when $C_O = C_I$, this results in

$$\bar{u}_1 = \frac{(c_{u_2} - c^+) d^{max}}{c_{u_2} + c^+ - c_{u_1}}. \tag{9.19}$$

If the retailer is able to order the maximal order quantity from the manufacturer $\bar{u}_1 \leq u_1^{max}$ and the maximal remaining quantity from the subcontractor $\bar{u}_2 \leq u_2^{max}$, the basestock policy is given by (9.16). This policy was also used in the numerical example.

Otherwise, if the retailer is not able to order the maximum order quantity from the manufacturer the basestock level follows from $\bar{u}_1 = u_1^{max}$ and solving $C_O = C_I$:

$$L = \frac{(c_{u_1} - c_{u_2}) u_1^{max} + c_{u_2} d^{max}}{c^+} - z, \quad \text{if } \bar{u}_1 > u_1^{max}.$$

Furthermore, if the retailer is not able to order $\bar{u}_2$ from the subcontractor the basestock level grows such that backlog is prevented. In this case $\bar{u}_2 = u_2^{max}$, therefore $\bar{u}_1 = d^{max} - u_2^{max}$ and from (9.15) it follows that

$$L = 2d^{max} - u_2^{max}, \quad \text{if } \bar{u}_2 < d^{max} - \bar{u}_1. \tag{9.20}$$

The feasible region of this network is given by

$$x_R + x_{R2} - x_d \geq 0, \tag{9.21}$$

$$z \geq d^{max} - u_2^{max}, \tag{9.22}$$

For completeness, optimal control for $c_{u_1} \geq c_{u_2}$ is presented here. This might occur due to special offers from the subcontractor. Logically, the maximal amount of products are ordered at the cheaper and faster subcontractor. Therefore, $\bar{u}_2 = \min(d^{max}, u_2^{max})$ and $\bar{u}_1 = \max(d^{max} - \bar{u}_2, 0)$.

In this section we have shown by example that using the method of constrained robust optimal control for a given supply network results in an optimal control policy. This policy was derived without any knowledge of policies a priori. However, a drawback of the method is the limitation to consider small networks. Due to DP, it involves an enormous amount of storage to even record the solution to a moderate complicated problem. For larger networks, the optimal solution can not be solved any more with DP, and a possible solution could be to use an approach of the cost-to-go function, e.g. quadratic function instead of piecewise linear function, where the solution remains close to the optimal solution. Another option is to consider MPC to derive controllers, see for instance [42].

9.3.5 Observers: State dependent control

Often control depends on multiple states. To achieve optimal performance, communication within the network is very important. Ideally, the retailer and suppliers should treat each other as strategic partners in the supply chain with flawless communication. However, as [29] and [30] also indicate, in many cases communication is a problem. For instance, the retailer can get inaccurate or no information about the supplier stock levels due to competition, organization or infrastructure. Therefore, we introduce the use of *observers* in the supply chain network to derive the control based on local information. An observer predicts unobservable or unmeasurable states based on observable/measurable parameters. For instance, in an airplane it could be difficult to measure the velocity of the plane directly while measuring time and position are easy. By measuring the time and position of the plane for a while, one can observe the velocity. Analogous for the supply network, by measuring the order and supply quantity long enough, the stock levels can be observed. We illustrate this approach by expanding the supply network from the previous example.

9.3.6 Example

In order to create a lack of state information due to communication issues we extend the example from Section 9.3.3 with warehouses that supply the manufacturer M and subcontractor S, as illustrated in Figure 9.15. Inventory levels of the retailer, manufacturer and subcontractor are denoted by x_R, x_M and x_S, respectively. Products ordered at the warehouses have a lead time of 2. Ordered products that have not been received by M or S are stored in intermediate states x_{M1} and x_{S1}, respectively. Resulting from the robust optimal control (9.8)-(9.9), both the manufacturer M and subcontractor S order their items with a basestock policy at a warehouse with infinite stock (∞) via channels u_M and u_S, respectively.

$$\begin{aligned} u_M(k) &= L_M - x_M(k) - x_{M1}(k), \\ u_S(k) &= L_S - x_S(k) - x_{S1}(k), \end{aligned}$$

where L_i are the basestock levels for x_i, $i \in \{M, S\}$.

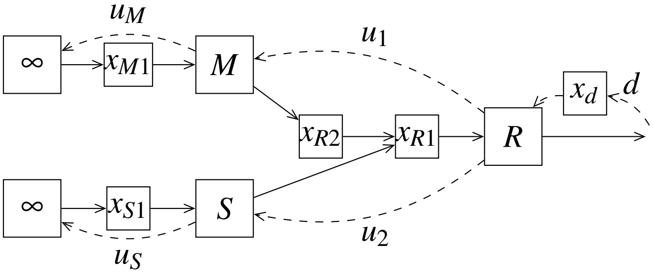


Figure 9.15: Supply network consisting of a retailer (R) and two suppliers (M and S). The retailer faces demand d, orders u_1 and u_2. The suppliers order via u_M and u_S from a warehouse with infinite stock (∞).

The retailer uses the policy

$$u_1(k) = \min[\max(L_R - z(k), 0), x_M(k)], \tag{9.23a}$$
$$u_2(k) = \min[\max(d^{max} - z(k), 0), x_S(k)], \tag{9.23b}$$

with basestock level L_R. The resulting controller for the retailer clearly depends on the stock levels of both suppliers. Therefore, proper communication between retailer and both suppliers is necessary to fulfill the customers demand.

Assuming that the retailer can get no information on the stock levels of both suppliers, the retailer faces a problem to ensure that the customers demand is met. We assume that the policies and base stock levels of the suppliers are known by the retailer. However, the actual stock levels are unknown which introduces a problem concerning the amount of products to order at the suppliers. To solve this problem we introduce observers, which are well-known in the control community, see for instance [75, 85]. First, consider the affine dynamics of the manufacturer M:

$$\begin{bmatrix} x_M(k+1) \\ x_{M1}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x_M(k) \\ x_{M1}(k) \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \end{bmatrix} u_1(k) + \begin{bmatrix} 0 \\ L_M \end{bmatrix}. \tag{9.24}$$

$$y(k) = \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} x_M(k) \\ x_{M1}(k) \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix} u_1(k), \tag{9.25}$$

where u_1 is the input from the retailer and y the output from the manufacturer.

A useful property to determine the states is *detectability*. A system is detectable when the states are asymptotically determined by the in- and outputs. Detectability of system (9.24)-(9.25) is addressed by considering the difference between the actual states and estimated states. This is denoted by the error dynamics

$$e(k) = |x(k) - \hat{x}(k)|,$$

where $\hat{x}$ indicates the estimated state. For system (9.24)-(9.25) the error dynamics are given by

$$e(k+1) = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} e(k). \tag{9.26}$$

It can be seen from the error dynamics for x_2 and x_{M1} in (9.26) that after two iterations the errors are zero. Therefore, system (9.24)-(9.25) is detectable. Furthermore, this indicates that the states x_M and x_{M1} can be exactly observed after two steps:

$$\begin{aligned} x_M(k+2) &= L_M - u_1(k+1) - u_1(k), \\ x_{M1}(k+2) &= u_1(k). \end{aligned}$$

Similar results are obtained for states x_S and x_{S1} of subcontractor S via input u_2:

$$\begin{aligned} x_S(k+2) &= L_S - u_2(k+1) - u_2(k), \\ x_{S1}(k+2) &= u_2(k). \end{aligned}$$

With this approach, the retailer can imply the control defined in (9.23) using the estimation of the states of both the manufacturer and subcontractor:

$$u_1(k) = \min[\max(L_R - z(k), 0), L_M - u_1(k-1) - u_1(k-2)], \tag{9.27a}$$
$$u_2(k) = \min[\max(d^{max} - z(k), 0), L_S - u_2(k-1) - u_2(k-2)]. \tag{9.27b}$$

The resulting-observer based controller (9.27) does not depend on communication anymore. The observers estimate the unknown supplier stock levels in two steps, whereafter the controller operates as desired.

9.3.7 Summary

First, we presented the use of techniques from robust optimal control to derive optimal control policies for discrete-time controlled dynamical systems with uncertainty. For the considered example of a supply network this led to the traditional, optimal ordering policy. However, this approach differs from other approaches by not assuming a certain strategy, or family of strategies, a priori. Also, instead of considering a fixed parameter setting, it is possible to derive optimal control for a network with unknown parameters. Due to use of Dynamic Programming, it involves an enormous amount of storage to even record the solution to a moderate complicated problem. For larger networks, the optimal solution can not be solved any more with Dynamic Programming, and a possible solution could be to use an approach of the cost-to-go function, e.g. quadratic function instead of piecewise linear function, where the solution remains close to the optimal solution. Another promising option from control theory is to consider the multi-variable control algorithm Model Predictive Control to derive controllers.

Second, the use of observers in supply networks is presented to derive explicit state-feedback control. The performance in a supply network for which the control of each state depends on more than the state itself can be non-optimal, due to a lack of information or incomplete information about other states in the network. Using observers in a observable/detectable supply network reduces the control to an explicit state-feedback control, i.e., there is no need for communication. An example is presented to illustrate this approach.

Chapter 10

Conclusions and recommendations

In this chapter, the conclusions from this thesis are summarized. Subsequently, some recommendations for further research on control and observer design of fluid flow networks are given.

10.1 Conclusions

This research is focussed on control and observer design for fluid flow switching servers. Control of switching servers is decoupled into an optimal periodic behavior problem and an optimal transient behavior problem. The optimal periodic behavior is the desired periodic reference trajectory, which leads to optimal long-term performance of the system. The optimal transient behavior is the trajectory that optimally steers the system towards the periodic behavior, if the system is removed from the optimal periodic behavior, e.g., due to disturbances.

In Chapter 3, a method is presented to derive the optimal periodic behavior for a two-queue switching server. The method can be applied in a variety of settings, e.g., setup times, setup costs and/or backlog, and generalizes a number of existing results. The optimal periodic behavior is formulated as a Linear Programming (LP) problem, or a Quadratic Programming (QP) problem and additional constraints can be easily implemented, e.g., bounds on queue contents and bounds on service and cycle times can be incorporated. For switching servers with multiple queues and where multiple queues can be served simultaneously, this method is extended in Chapter 4. For these servers, multiple sequences exist. For each sequence the optimal periodic behavior is derived and the sequence with the best performance yields the optimal periodic behavior. Reasonable bounds on the number of groups in a sequence and the number of service times allowed for a single queue in a sequence are imposed to limit the number of possible sequences. It is shown that allowing multiple service periods for a queue in a cycle can have a substantial (positive) effect on the performance of the system. In Chapter 5, optimal periodic behavior for a network of switching servers with instant transportation between queues is presented. Due to the network topology, arrival rates at queues can be nonconstant. By aggregation of queues, the problem of deriving optimal periodic behavior can also be formulated as a QP problem. The number of sequences quickly grows by relaxing the bounds on number of groups and total number of service periods per queue. Since each sequence is optimized separately, the computational efforts increase enormously. Therefore, the presented approach seems to be suitable for moderate networks of switching servers.

Given a switching server and the optimal periodic behavior, a method is presented in Chapter 6 that derives the optimal transient behavior from any initial state of the system. For the unconstrained system without backlog, the optimal control policy is derived using switching curves.

In practice, the complete state of the system is hardly ever completely known. However, control policies might require the complete state of the system to perform as is desired. Hence, observer design, i.e., reconstructing the unknown states based on measured information, is important for implementing the control policy. For multi-queue switching servers with a clearing policy, we designed observers which recover the state of the system in Chapter 7. By measuring the input and the limited information from the system, observers are designed in a stepwise approach. The design principles for the single server are also used to design an observer for a specific network with a specific control policy, as a step towards observer design for networks of switching servers, presented in Chapter 8. Here, the cyclic evolution of modes allows the observer to reconstruct the state of the network.

In Chapter 9, some related problems are addressed. For a network with non-zero transportation times, a heuristic is presented to derive a good service schedule based on the optimal periodic behavior of the individual switching servers. Also, switching servers with discrete objects and poisson distributed inter-arrival times are considered. For this system, we present an approximation of the average queue content during a cycle and derive service schedules using this approximation. The resulting schedule is compared to the schedule derived for the fluid flow model and it is shown that it performs better for systems with high loads. Finally, it is shown that by using techniques from robust optimal control, without assuming a certain strategy a priori, the optimal policy can be derived for supply networks. It is also shown that by the use of observers, for detectable or observable supply networks, unknown states can be reconstructed.

10.2 Recommendations

In this thesis, control and observer design of fluid flow switching systems are presented. There are many possibilities and directions for further research. Some recommendations for further research are discussed below.

The presented methods to derive the optimal periodic and optimal transient behavior of a system consists of generating all feasible sequences and solving an optimization problem for each sequence. For large systems, this results in a vast amount of sequences, which all need to be optimized, which is computationally expensive. Hence, we proposed bounds on the number of groups in a sequence and a maximal number of service periods per queue in a sequence to embank the number of feasible sequences. To avoid this 'brute force' optimization of all sequences, it is interesting to look into methods to derive a class of sequences, or subset of all feasible sequences, which governs the sequence resulting in the optimal behavior. For some sequences, or classes of sequences, it might be possible to show that these are non-optimal, and therefore can be omitted in the optimization step. Also, heuristics can be developed that minimize the number of sequences to optimize, which result in a good (not necessarily optimal) service schedule.

For networks of switching servers, as also touched upon in Section 9.1, the optimal control problem for networks with transportation times remains open. A step towards solving this problem could be to use 'negative' setup times to compensate for the transportation times, i.e., before the end of a service period of a queue a conflicting queue can start its service period. This requires a reformulation of the sequence composition and constraints. Also, the problem of optimal transient behavior for networks of switching servers has not been discussed in this thesis. This problem is similar to the transient behavior problem for multi-queue switching servers and therefore it is expected that this problem can be solved similarly.

In this thesis, fluid flow switching servers are considered. However, for systems with low-volume discrete objects, the fluid flow approximation is inappropriate. Hence, the problem of control and observer design for such discrete systems remains a topic for further research. The results presented in this thesis might serve as a guideline towards optimal control and observer design.

The presented methods use constant arrival and service rates for deriving the optimal behavior. Variations are not taken into account, except for a brief discussion on Poisson arrivals in Section 9.2. The question remains whether the schedules are still optimal in a stochastic environment. In order to derive the controllers for a real-life environment, stochastic processes in the control and observer design needs to be looked into.

The results from the observer design for a single switching server (Chapter 7) lead to various questions. First of all, it would be of interest to take the positivity of the state variable into account leading to observers that always create positive state estimates as well (positive observers). Some first hints were already provided in this direction. In addition, it is relevant to formulate necessary and sufficient conditions in terms of the data of the original system when the proposed design is indeed successful and how this relates to fundamental observability and detectability properties. Furthermore, systems with disturbances, such as, e.g., server failure or stochastic arrival or service processes, are not taken into account. As in practice the arrival or service rates may fluctuate or servers can break down, the design of observers for systems with stochastic behavior is of interest for future work. For observer design of networks of switching servers, for which a first attempt is presented in Chapter 8, also questions remain which can be valuable for further research. Some of these questions are identical to the questions for the single server observer design problem. First of all, it is shown that for a specific network and with a specific policy, an observer can be designed by estimating all feasible states and eliminating infeasible states. Convergence to the actual state of the network is presented via simulation, proving the convergence is a topic for further research. Also, the elimination of feasible states is based on perfect timing of the next expected visible event. Therefore, for a network with disturbances, the observer may not be able to reconstruct the network's state. Hence, the design of observers for networks with stochastic behavior is of interest for future work. Second, it is relevant to investigate the applicability to other networks or other policies. For the presented observer design approach, a cyclic evolution of modes is required by the observer and it is also required that the visible events are created by an unique trajectory of the system. In addition, it is also of importance to formulate necessary and sufficient conditions for the observer design and their relation to observability and detectability properties. Finally, it is interesting to investigate to what extent the observer design principles put forward in this thesis can be applied to more general classes of hybrid systems. As such, the provided ideas might be fruitfully exploited into various future research directions.

A further step in the modeling and control framework is the implementation of the proposed controllers on actual systems.

References

[1] ALESSANDRI, A., BAGLIETTO, M., BATTISTELLI, G., AND ZAVALA, V. Advances in moving horizon estimation for nonlinear systems. In *Decision and Control (CDC), 2010 49th IEEE Conference on* (Dec 2010), pp. 5681–5688.

[2] ALESSANDRI, A., AND COLETTA, P. Design of Luenberger observers for a class of hybrid linear systems. In *Proc. of Hybrid Systems: Computation and Control* (Rome, 2001), vol. 2034 of *Lecture Notes in Computer Science*, Springer, pp. 7–18.

[3] ALESSANDRI, A., AND COLETTA, P. Design of observers for switched discrete-time linear systems. In *American Control Conference. Proceedings of the 2003* (June 2003), vol. 4, pp. 2785–2790.

[4] ALLSOP, R. E. SIGSET: A computer program for calculating traffic capacity of signal-controlled road junctions. *Traffic Engineering & Control 12* (1971), 58–60.

[5] ALLSOP, R. E. SIGCAP: A computer program for assessing the traffic capacity of signal-controlled road junctions. *Traffic Engineering & Control 17*, 8-9 (1976), 338–341.

[6] ALUR, R., AND DILL, D. L. A theory of timed automata. *Theoretical Computer Science 126* (1994), 183–235.

[7] BABAALI, M. Observability of switched linear systems in continuous time. In *Lecture Notes in Computer Sciences* (2005), Springer-Verlag, pp. 103–117.

[8] BABAALI, M., AND EGERSTEDT, M. Observability for switched linear systems. In *Hybrid Systems: Computation and Control*, vol. 2623 of *Lecture Notes in Computer Science. Hybrid Systems: Computation and Control.* Springer, 2004.

[9] BAI, S., AND ELHAFSI, M. Transient and steady-state analysis of a manufacturing system with setup changes. *Journal of Global Optimization 8* (1996), 349–378.

[10] Balluchi, A., Benvenuti, L., Di Benedetto, M. D., and Sangiovanni-Vincentelli, A. L. Design of observers for hybrid systems. In *Hybrid Systems: Computation and Control*, C. Tomlin and J. Greenstreet, Eds., vol. 2289 of *Lecture Notes in Computer Science*. Springer-Verlag, Stanford, CA, 2002, pp. 76–89.

[11] Balluchi, A., Di Benedetto, M. D., Benvenuti, L., and Sangiovanni-Vincentelli, A. L. Observability for hybrid systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control* (2003), vol. 2, pp. 1159–1164.

[12] Banks, J., and Dai, J. G. Simulation studies of multiclass queueing networks. *IIE Transactions 29*, 3 (1997), 213–219.

[13] Bara, G. I., Daafouz, J., Kratz, F., and Ragot, J. Parameter dependent state observer design for affine LPV systems. *International Journal of Control 74*, 16 (2001), 1601–1611.

[14] Baras, J. S., Ma, D. J., and Makowski, A. M. K competing queues with geometric service requirements and linear costs: The μc-rule is always optimal. *Systems & Control Letters 6*, 3 (1985), 173 – 180.

[15] Bemporad, A., Ferrari-Trecate, G., and Morari, M. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control 45*, 10 (2000), 1864–1876.

[16] Bertrand, J. W. M., Wortmann, J. C., and Wijngaard, J. *Production Control: A Structural and Design Oriented Approach.* Elsevier, Amsterdam, The Netherlands, 1990.

[17] Bertsekas, D. P. *Dynamic Programming and Optimal Control.* Athena Scientific, 2001.

[18] Bielli, M., and Reverberi, P. New operations research and artificial intelligence approaches to traffic engineering problems. *European Journal of Operational Research 92*, 3 (1996), 550–572.

[19] Boccadoro, M., and Valigi, P. A switched system model for the optimal control of two symmetric competing queues with finite capacity. In *Current Trends in Nonlinear Systems and Control*, L. Menini, L. Zaccarian, and C. Abdallah, Eds., Systems and Control: Foundations and Applications. Birkhauser Boston, 2006, pp. 455–474.

[20] Boon, M. A. A., Adan, I. J. B. F., Winands, E. M. M., and Down, D. G. Delays at signalised intersections with exhaustive traffic control. *Probability in the Engineering and Informational Sciences 26*, 03 (2012), 337–373.

[21] BRAMSON, M. Stability of queueing networks. *Probability Surveys 5* (2008), 169–345.

[22] BROEK, M. S. VAN DEN. Traffic signals: Optimizing and analyzing traffic control systems. MSc thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science, Eindhoven, The Netherlands, 2004.

[23] BRUNO, G., AND IMPROTA, G. Individual controlled junctions: Optimal design. *European Journal of Operational Research 71*, 2 (1993), 222 – 234.

[24] BUYUKKOC, C., VARAIYA, P., AND WALRAND, J. The μc-rule revisited. *Advances in Applied Probability 17* (1985), 237 Ű– 238.

[25] CAMLIBEL, M. K., PANG, J. S., AND SHEN, J. Conewise linear systems: non-zenoness and observability. *SIAM J. Control Optimizaton 45* (2006), 1769–1800.

[26] CHASE, C., AND RAMADGE, P. On real-time scheduling policies for flexible manufacturing systems. *Automatic Control, IEEE Transactions on 37*, 4 (April 1992), 491–496.

[27] CHEN, T., AND FRANCIS, B. A. *Optimal sampled-data control systems.* Springer-Verlag, London, 1995.

[28] COLLINS, P., AND SCHUPPEN, J. H. VAN. *Observability of Piecewise-Affine Hybrid Systems*, vol. 2993 of *Lecture Notes in Computer Science. Hybrid Systems: Computation and Control.* Springer Berlin - Heidelberg, 2004, pp. 265–279.

[29] CUTTING-DECELLE, A. F., DAS, B. P., YOUNG, R. I., CASE, K., RAHIMIFARD, S., ANUMBA, C. J., AND BOUCHLAGHEM, N. M. Building supply chain communication systems: A review of methods and techniques. *Data Science Journal 5* (2006), 1–23.

[30] CUTTING-DECELLE, A. F., YOUNG, B. I., DAS, B. P., CASE, K., RAHIMIFARD, S., ANUMBA, C. J., AND BOUCHLAGHEM, D. M. A review of approaches to supply chain communications: From manufacturing to construction. *Electronic Journal of Information Technology in Construction 12* (2007), 73–102.

[31] DAGANZO, C. F. *A Theory of Supply Chains.* Springer, Heidelberg, Germany, 2003.

[32] DE SCHUTTER, B. Optimizing acyclic traffic signal switching sequences through an extended linear complementarity problem formulation. *European Journal of Operational Research 139*, 2 (2002), 400–415.

[33] DEL GAUDIO, M., MARTINELLI, F., AND VALIGI, P. A scheduling problem for two competing queues with finite capacity and non-negligible setup times. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on* (2001), vol. 3, pp. 2355–2360 vol. 3.

[34] DUISTERS, K. S. Formulating and testing an algorithm for fixed time control of traffic intersections. Internship report MN 420734, Eindhoven University of Technology, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2013.

[35] EEKELEN, J. A. W. M. VAN. *Modelling and control of discrete event manufacturing flow lines*. PhD thesis, Eindhoven, University of Technology, 2008.

[36] EEKELEN, J. A. W. M. VAN, LEFEBER, E., AND ROODA, J. E. Feedback control of 2-product server with setups and bounded buffers. In *Proceedings of the 2006 American Control Conference* (2006), pp. 544–549.

[37] EEKELEN, J. A. W. M. VAN, LEFEBER, E., AND ROODA, J. E. Feedback control of 2-product server with setups and bounded buffers. In *2006 American Control Conference* (New York, NY, USA, 2006), vol. 1-12 of *Proceedings of the American Control Conference*, IEEE, pp. 544–549.

[38] EEKELEN, J. A. W. M. VAN, LEFEBER, E., AND ROODA, J. E. State feedback control of switching server flowline with setups. In *Proceedings of the American Control Conference* (New York City, NY, USA, 2007), pp. 3618–3623.

[39] ELHAFSI, M., AND BAI, S. X. Optimal production and setup control of a dynamic two-product manufacturing system: Analytical solution. *Mathematical and Computer Modelling 24*, 3 (1996), 57 – 78.

[40] FERRARI-TRECATE, G., MIGNONE, D., AND MORARI, M. Moving horizon estimation for hybrid systems. *IEEE Transactions on Automatic Control 47* (2002), 1663–1676.

[41] GALLIVAN, S., AND HEYDECKER, B. Optimising the control performance of traffic signals at a single junction. *Transportation Research Part B: Methodological 22*, 5 (1988), 357–370.

[42] GARCIA, C. E., PRETT, D. M., AND MORARI, M. Model predictive control: Theory and practice: A survey. *Automatica 25*, 3 (1989), 335 – 348.

[43] GOEBEL, R., SANFELICE, R., AND TEEL, A. R. Hybrid dynamical systems. *IEEE Control Systems Magazine 29*, 2 (2009), 28–93.

[44] HADDAD, J., DE SCHUTTER, B., MAHALEL, D., IOSLOVICH, I., AND GUTMAN, P. O. Optimal Steady-State Control for Isolated Traffic Intersections. *Automatic Control, IEEE Transactions on 55*, 11 (2010), 2612–2617.

[45] HADDAD, J., GUTMAN, P.-O., IOSLOVICH, I., AND MAHALEL, D. Discrete dynamic optimization of N-stages control for isolated signalized intersections. *Control Engineering Practice 21*, 11 (2013), 1553 – 1563.

[46] HE, Z., CHEN, Y., SHI, J., HAN, X., AND WU, X. Steady-state control for signalized intersections modeled as switched server system. In *American Control Conference (ACC), 2013* (2013), pp. 842–847.

[47] HEEMELS, W. P. M. H., DE SCHUTTER, B., LUNZE, J., AND LAZAR, M. Stability analysis and controller synthesis for hybrid dynamical systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 368*, 1930 (2010), 4937–4960.

[48] HEYMAN, D. P., AND STIDHAM, JR. S. The relation between customer and time averages in queues. *Operations Research 28*, 4 (1980), 983–994.

[49] HOFRI, M., AND ROSS, K. W. On the optimal control of two queues with server setup times and its analysis. *SIAM Journal on Computing 16*, 2 (1987), 399–420.

[50] HOSTETTER, G. H. Ongoing deadbeat observers for linear time-varying systems. In *American Control Conference* (1982), pp. 1099–1101.

[51] HU, J. Discrete-time linear periodically time-varying systems: Analysis, realization and model reduction. Master's thesis, Rice University, July 2003.

[52] HUMES, C. JR. A regulator stabilization technique: Kumar-seidman revisited. *Automatic Control, IEEE Transactions on 39*, 1 (1994), 191–196.

[53] IMBASTARI, V., MARTINELLI, F., AND VALIGI, P. An optimal scheduling problem for a system with finite buffers and non-negligible setup times and costs. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on* (dec. 2002), vol. 1, pp. 1156– 1161 vol. 1.

[54] IMPROTA, G., AND CANTARELLA, G. E. Control system design for an individual signalized junction. *Transportation Research Part B: Methodological 18*, 2 (1984), 147–167.

[55] IOSLOVICH, I., HADDAD, J., GUTMAN, P., AND MAHALEL, D. Optimal traffic control synthesis for an isolated intersection. *Control Engineering Practice 19*, 8 (2011), 900 – 911.

[56] IRANI, S., AND LEUNG, V. Scheduling with conflicts, and applications to traffic signal control. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms* (Philadelphia, PA, USA, 1996), SODA '96, Society for Industrial and Applied Mathematics, pp. 85–94.

[57] IVANOV, D., DOLGUI, A., AND SOKOLOV, B. Applicability of optimal control theory to adaptive supply chain planning and scheduling. *Annual Reviews in Control 36*, 1 (Apr. 2012), 73–84.

[58] JULOSKI, A. L., HEEMELS, W. P. M. H., AND WEILAND, S. Observer design for a class of piecewise linear systems. *International Journal of Robust and Nonlinear Control 17*, 15 (2007), 1387–1404.

[59] KIMEMIA, J., AND GERSHWIN, S. B. An algorithm for the computer control of a flexible manufacturing system. *IIE Transactions 15*, 4 (1983), 353–362.

[60] KOK, A. G. DE, AND FRANSOO, J. C. *Handbooks in Operation Research and Management Science*, vol. 11. Elsevier, 2003, ch. Planning supply chain operations: definition and comparison of planning concepts, pp. 597–676.

[61] KOK, A. G. DE, AND GRAVES, S. C. *Supply Chain Management: Design, Coordination and Operation, Handbooks in Operations Research and Management Science*, vol. 11. Elsevier, B. V, 2003.

[62] KUMAR, P. R., AND SEIDMAN, T. I. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *Automatic Control, IEEE Transactions on DOI - 10. 1109/9. 50339 35*, 3 (1990), 289–298.

[63] LAN, W.-M., AND OLSEN, T. L. Multiproduct systems with both setup times and costs: Fluid bounds and schedules. *Operations Research 54*, 3 (2006), 505–522.

[64] LAUMANNS, M., AND LEFEBER, E. Robust optimal control of material flows in demand-driven supply networks. *Physica A: Statistical Mechanics and its Applications 363*, 1 (Apr. 2006), 24–31.

[65] LEFEBER, A. A. J., AND ROODA, J. E. Optimal behavior for the kumar-seidman network of switching servers. In *Proceedings of the 6th EUROMECH Nonlinear Dynamics Conference (ENOC08)* (2008).

[66] LEFEBER, E., LAEMMER, S., AND ROODA, J. E. Optimal control of a deterministic multiclass queuing system for which several queues can be served simultaneously. *Systems & Control Letters 60*, 7 (2011), 524 – 529.

[67] LEFEBER, E., AND ROODA, J. E. Controller design for switched linear systems with setups. *Physica A: Statistical Mechanics and its Applications 363*, 1 (2006), 48–61.

[68] LEFEBER, E., AND ROODA, J. E. Controller design for flow networks of switched servers with setup times: The kumar-seidman case as an illustrative example. *Asian Journal of Control 10*, 1 (Jan. 2008), 55–66.

[69] LITTLE, J. D. C. A proof for the queuing formula: $L = \lambda W$. *Operations Research 9*, 3 (1961), 383–387.

[70] LIU, Z., NAIN, P., AND TOWSLEY, D. On optimal polling policies. *Queueing Systems 11* (1992), 59–83.

[71] LUENBERGER, D. G. Observing state of linear system. *IEEE Transactions On Military Electronics 8*, 2 (1964), 74–80.

[72] LUNZE, J., AND LAMNABHI-LAGARRIGUE, F. *The HYCON Handbook of Hybrid Systems Control: Theory, Tools, Applications.* Cambridge University Press, Cambridge, 2009.

[73] MARTINELLI, F., AND VALIGI, P. Dynamic scheduling for a single machine system under different setup and buffer capacity scenarios. *Asian Journal of Control 6*, 2 (jun 2004), 229–241.

[74] MOHSEN HOSSEINI, S., AND OROOJI, H. Phasing of traffic lights at a road junction. *Applied Mathematical Sciences 3*, 29-32 (2009), 1487–1492.

[75] O'REILLY, J. *Observers for linear systems*. Academic Press: Mathematics in Science & Engineering, London, 1983.

[76] ORTEGA, M., AND LIN, L. Control theory applications to the production-inventory problem: a review. *International Journal of Production Research 42*, 11 (June 2004), 2303–2322.

[77] OZVEREN, C. M., AND WILLSKY, A. S. Observability of discrete event dynamic systems. *Automatic Control, IEEE Transactions on 35*, 7 (1990), 797–806.

[78] PAPAGEORGIOU, M., DIAKAKI, C., DINOPOULOU, V., KOTSIALOS, A., AND WANG, Y. Review of road traffic control strategies. *Proceedings of the IEEE 91*, 12 (Dec 2003), 2043–2067.

[79] PERKINS, J. R., HUMES, C. JR., AND KUMAR, P. R. Distributed scheduling of flexible manufacturing systems: stability and performance. *Robotics and Automation, IEEE Transactions on 10*, 2 (1994), 133–141.

[80] PERKINS, J. R., AND KUMAR, P. R. Stable, distributed, real-time scheduling of flexible manufacturing/assembly/diassembly systems. *Automatic Control, IEEE Transactions on 34*, 2 (1989), 139–148.

[81] PETTERSON, S. Switched state jump observers for switched systems. In *Proceedings of the IFAC World Congress* (Prague, Czech Republic, 2005).

[82] PINEDO, M. L. Economic lot scheduling. In *Planning and Scheduling in Manufacturing and Services*. Springer New York, 2009, pp. 143–171.

[83] RAMI, M. A., AND TADEO, F. Positive observation problem for linear discrete positive systems. In *Proceedings of the 45th IEEE Conference on Decision and Control* (San Diego, CA, USA, 2006), pp. 4729–4733.

[84] REIMAN, M. I., AND WEIN, L. M. Dynamic scheduling of a two-class queue with setups. *Operations Research 46*, 4 (Apr. 1998), 532–547.

[85] RUGH, W. J. *Linear System Theory (2nd ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[86] SAVKIN, A. Optimal distributed real-time scheduling of flexible manufacturing networks modeled as hybrid dynamical systems. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on* (Dec 2003), vol. 5, pp. 5468–5471.

[87] SCHINKEL, M., HEEMELS, W. P. M. H., AND JULOSKI, A. L. State estimation for systems with varying sampling rate. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on* (Dec 2003), vol. 1, pp. 391–392.

[88] SERAFINI, P., AND UKOVICH, W. A mathematical model for the fixed-time traffic control problem. *European Journal of Operational Research 42*, 2 (Sept. 1989), 152–165.

[89] STIDHAM, S. A last word on $L = \lambda W$. *Operations Research 22* (1974), 417 – 421.

[90] STOFFERS, K. E. Scheduling of traffic lights. a new approach. *Transportation Research 2*, 3 (1968), 199–234.

[91] TAKAGI, H. Queueing analysis of polling models: progress in 1990-1994. In *Frontiers in queueing*, J. H. Dshalalow, Ed. CRC Press, Inc., Boca Raton, FL, USA, 1997, pp. 119–146.

[92] TAN, B., AND GERSHWIN, S. B. Production and subcontracting strategies for manufacturers with limited capacity and volatile demand. *Annals of Operations Research 125* (2004), 205–232.

[93] TANWANI, A., SHIM, H., AND LIBERZON, D. Observability implies observer design for switched linear systems. *Proceedings of the 2011 Hybrid Systems: Computation and Control* (2011), 3–12.

[94] TANWANI, A., SHIM, H., AND LIBERZON, D. Observability for switched linear systems: Characterization and observer design. *Automatic Control, IEEE Transactions on 58*, 4 (April 2013), 891–904.

[95] VLEUTEN, J. W. C. M. VAN DER. Networks of signalized traffic intersections, optimization and implementation. MSc thesis MN 420773, Eindhoven University of Technology, the Netherlands, 2014.

[96] WEBSTER, F. V. *Traffic Signal Settings.* Road Research Technical Paper No. 39, London, England, 1958.

[97] WONG, C. K., AND HEYDECKER, B. G. Optimal allocation of turns to lanes at an isolated signal-controlled junction. *Transportation Research Part B: Methodological 45*, 4 (2011), 667 – 681.

[98] WONG, C. K., AND WONG, S. C. Lane-based optimization of signal timings for isolated junctions. *Transportation Research Part B: Methodological 37*, 1 (2003), 63 – 84.

[99] WOUW N. VAN DE, AND PAVLOV, A. Tracking and synchronisation for a class of PWA systems. *Automatica 44*, 11 (2008), 2909–2915.

[100] YANG, Y., MAO, B., CHEN, S., LIU, S., AND LIU, M. Effect of bus rapid transit signal priority effect on traffic flow. *Shenzhen University Science and Engineering 30*, 1 (2013), 91–97.

[101] YECHIALI, U. Analysis and control of polling systems. In *Performance Evaluation of Computer and Communication Systems*, L. Donatiello and R. Nelson, Eds., vol. 729 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1993, pp. 630–650.

[102] ZUZARTE, T. I. M. *Synthesis of sequences for traffic signal controllers using techniques of the theory of graphs*. PhD thesis, University of Oxford, 1977.

[103] ZWIETEN, D. A. J. VAN. Optimal process cycle and feedback control of a 3-product workstation with setups. B. Sc thesis SE 420512, Eindhoven University of Technology, the Netherlands, 2007.

[104] ZWIETEN, D. A. J. VAN, LEFEBER, E., AND ADAN, I. J. B. F. Optimal periodical behavior of an isolated traffic intersection with pretimed signals. *Submitted for publication*.

[105] ZWIETEN, D. A. J. VAN, LEFEBER, E., AND ADAN, I. J. B. F. Optimal periodical scheduling of multi-class fluid flow networks. *Submitted for publication*.

[106] ZWIETEN, D. A. J. VAN, LEFEBER, E., AND ADAN, I. J. B. F. Optimal steady-state and transient trajectories of a two queue switching server. *Submitted for publication*.

[107] ZWIETEN, D. A. J. VAN, LEFEBER, E., AND ADAN, I. J. B. F. Optimal periodical behavior of a multiclass fluid flow network. In *Proceedings of the 6th IFAC Conference on Management and Control of Production and Logistics (IFAC-PapersOnline)* (2013), vol. 6, pp. 95–100.

[108] ZWIETEN, D. A. J. VAN, LEFEBER, E., AND ADAN, I. J. B. F. Optimal steady-state and transient trajectories of a two queue switching server. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools* (2013).

[109] ZWIETEN, D. A. J. VAN, LEFEBER, E., ADAN, I. J. B. F., AND ROODA, J. E. Control of supply networks by robust optimal control and using observers. In *Proceedings of the 7th IFAC Conference on Manufacturing Modelling, Management, and Control (IFAC-PapersOnline)* (2013), pp. 1584–1589.

[110] Zwieten, D. A. J. van, Lefeber, E., and Heemels, W. P. M. H. Observer design for a class of piecewise affine hybrid systems. In *HSCC 2013 – Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control* (2012), pp. 153–162.

Nederlandse samenvatting

In een wereld waar de complexiteit van systemen alleen maar groeit is de vraag om besturingsstrategieën overal te vinden. In dit onderzoek bekijken we deze vraag met betrekking tot systemen van schakelende werkstations. In een geschakeld werkstation worden meerdere wachtrijen bedient en schakelen tussen de bediening van wachtrijen kan tijd en/of geld kosten. Dit soort systemen komen overal voor, zoals in productiesystemen, voedselverwerkingsfaciliteiten of computercommunicatienetwerken. Geschakelde werkstations kunnen ook de dynamiek van alledaagse situaties beschrijven, denk aan verkeersstromen op kruispunten met verkeerslichten of wachtrijen in ziekenhuizen. Om meer inzicht in de (basis) dynamiek van schakelende werkstations te verkrijgen, beschouwen we een werkstation als een vloeistofmodel. In dit model komt vloeistof met een constante snelheid aan, wordt de vloeistof in de wachtrij met een constante snelheid bediend en zijn er geen verstoringen. Het is normaal om een continue stroom van objecten aan te nemen als de discrete objecten in een grote hoeveelheid aankomen. Systemen waar de objecten in een kleine hoeveelheid aankomen vallen buiten het bereik van dit onderzoek. Dit komt doordat modellen voor discrete objecten hier beter bij passen, want elk individueel object kan een grote invloed hebben op het systeem.

In dit onderzoek is het besturen van schakelende werkstations opgesplitst in het bepalen van optimaal periodiek gedrag en het bepalen van optimaal transiënt gedrag. Het optimale periodieke gedrag is het gewenste referentietraject dat leidt tot een optimale prestatie van het systeem over de lange termijn. Het optimale transiënte gedrag is het traject dat het systeem optimaal naar het periodieke gedrag stuurt. Dit kan nodig zijn als het systeem van het periodieke gedrag verwijderd is, bijvoorbeeld door onderhoudswerkzaamheden of bedieningsprioriteiten.

We presenteren een methode om het optimale periodieke gedrag van een enkel schakelend werkstation met twee wachtrijen te bepalen. Een analytische afleiding van het optimale periodieke gedrag voor schakelende werkstations met meer dan twee wachtrijen of met beperkingen op de wachtrijlengtes of bedieningsperiodes wordt snel te complex, als het al mogelijk is om dit af te leiden. Daarom formuleren we het probleem van optimaal periodiek gedrag als een lineair programmerings- (LP) of kwadratisch programmeringsprobleem (QP). Dit geldt voor systemen met omsteltijden, omstelkosten en/of achterstallig werk. Deze methode is flexibel met betrekking tot verschillende prestatiecriteria. Ook kunnen additionele systeem-beperkingen makkelijk toegevoegd worden, zoals beperkingen op de wachtrijlengtes of bedienings- en cyclustijden

Vervolgens is deze methode uitgebreid om het optimale periodieke gedrag van een enkel geschakeld werkstation met meer dan twee wachtrijen te bepalen. In deze systemen kunnen meerdere wachtrijen tegelijkertijd bediend worden met een wachtrij-afhankelijke snelheid. Vandaar kunnen kruispunten met verkeerslichten, waar meerdere voertuigstromen op hetzelfde moment een groen licht mogen krijgen, ook gemodelleerd worden. De methode bestaat uit twee opeenvolgende stappen. Eerst worden alle uitvoerbare sequenties gegenereerd, omdat voor systemen met meer dan twee wachtrijen de volgorde van het bedienen van de wachtrijen onbekend is. Als tweede worden de optimale bedieningstijden per sequentie berekend. We staan toe dat een wachtrij meerdere bedieningsperiodes heeft in een cyclus. Door middel van voorbeelden laten we zien dat dit een substantieel (positief) effect kan hebben op het resultaat.'

Voor een netwerk van schakelende werkstations, waar objecten via vastgestelde routes naar (verscheidene) werkstations in het netwerk stromen presenteren we een zelfde aanpak om het optimale periodieke gedrag te bepalen. Het verschil met een enkel schakelend werkstation is dat de aankomstsnelheid van stromen niet meer altijd constant is. In het netwerk beweegt de vloeistof tussen de wachtrijen, wat leidt tot een stuksgewijs constante aankomstsnelheid bij wachtrijen (de stuksgewijze vertreksnelheid van een wachtrij in een werkstation is de aankomstsnelheid van de wachtrij verderop langs de route). Daarom zijn bedieningsperiodes verdeeld in meerdere fasen en gebruiken we aggregatie van wachtrijen om de wachtrijlengtes in het netwerk te bepalen. Voor de aggregatie van wachtrijen is aangenomen dat er geen transporttijden tussen werkstations zijn. Na bediening van een stroom is de vloeistof onmiddellijk aanwezig in de volgende wachtrij. Dan kan de optimalisatie van elke sequentie geformuleerd worden als een QP-probleem en optimaal periodiek gedrag voor netwerken met schakelende werkstations bepaald worden.

Vervolgens is het optimale transiënte gedrag van een enkel geschakeld werkstation onderzocht, met een van te voren bepaald optimaal periodiek gedrag. We presenteren een methode dat het systeem optimaal naar het periodieke gedrag stuurt. Voor een willekeurige begintoestand is het optimale transiënte probleem geformuleerd als een QP-probleem. Voor een werkstation met twee wachtrijen is het systeem met én zonder achterstallig werk apart beschouwd. Door het combineren van omschakelpunten, systeemtoestanden voor welke het werkstation omschakelt naar de andere wachtrij, kan de optimale besturingsstrategie afgeleid worden voor een systeem zonder beperkingen en zonder achterstallig werk. Deze strategie is uitgedrukt in omschakelcurves: als een punt op deze curve bereikt wordt moet het systeem omschakelen naar de andere wachtrij. De omschakelpunten zijn afhankelijk van beperkingen op wachtrijlengtes of bedieningsperiodes. Hierdoor zijn er voor systemen met deze beperkingen geen omschakelcurves. Deze methode kan uitgebreid worden naar werkstations met meerdere wachtrijen. Alleen is voor deze werkstations de volgorde van het bedienen van de wachtrijen (gewoonlijk) niet bekend. Vergelijkbaar met de methode voor optimaal periodiek gedrag kunnen alle sequenties gegenereerd worden en voor elke sequentie kan de optimale oplossing gevonden worden. Dan geeft de beste oplossing van alle sequenties het optimale transiënte gedrag.

De besturingen die volgen uit de voorgenoemde methodes zijn allen centrale besturingen en bepalen het schakelgedrag van elk werkstation, gebaseerd op de toestand van het gehele systeem (globale toestand). Voor niet-kunstmatige systemen, zoals kruispunten met verkeerslichten, is in de werkelijkheid de globale toestand bijna nooit beschikbaar. Dit maakt het ontwerpen van waarnemers van cruciaal belang. Een waarnemer bepaalt een goede schatting voor de globale toestand van het systeem, gebaseerd op de in- en output van het systeem. Als een eerste stap richting een waarnemerontwerp voor een generiek geschakeld werkstation presenteren wij een waarnemerontwerp voor een geschakeld werkstation, dat werkt volgens een bepaalde besturingsstrategie. Deze systemen, welke relevant zijn in de context van productie- en verkeersapplicaties, zijn onderdeel van een speciale klasse van stuksgewijs affine hybride systemen. Ondanks het feit dat alle subsystemen niet waarneembaar zijn en niet alle gebeurtenissen zichtbaar zijn, kan een waarnemer in continue tijd geconstrueerd worden. Deze garandeert dat de schatting convergeert naar de toestand van het werkelijke systeem. Het basisidee is om het systeem te evalueren op zichtbare gebeurtenissen. Hiervoor kan via de standaardtechnieken uit de regeltechniek een waarnemer in discrete tijd ontworpen worden. Deze waarnemer wordt vervolgens als blauwdruk voor de waarnemer in continue tijd gebruikt. Naast de systeemdynamica worden additionele ‘wacht’modes toegeschreven aan de ontwerpen. Uiteindelijk laten we zien dat deze principes resulteren in een succesvol waarnemerontwerp.

Voor het Kumar Seidman netwerk en een specifieke besturing is een soortgelijke aanpak gebruikt om een waarnemer te ontwerpen. Dit is een eerste poging in de richting van een waarnemerontwerp voor netwerken van schakelende werkstations. Ondanks een minimale hoeveelheid gemeten informatie is een waarnemer ontworpen, welke naar de huidige toestand van het netwerk convergeert. Deze aanpak is drievoudig. Eerst is het (periodieke) schakelsysteem bepaald, gebaseerd op de besturing en de dynamica van het netwerk. Ten tweede wordt het netwerk geëvalueerd op zichtbare gebeurtenissen en wordt de dynamica tussen deze gebeurtenissen afgeleid. Ten derde evalueert de waarnemer alle mogelijke voorspelde toestanden op de zichtbare gebeurtenissen. Door eliminatie van onmogelijke toestanden blijft er uiteindelijk een enkele geschatte toestand voor het netwerk over. Door middel van simulatie is aangetoond dat deze toestand convergeert naar de actuele toestand van het netwerk.

Uiteindelijk worden enkele problemen gerelateerd aan besturing en waarnemerontwerp voor schakelende werkstations besproken. Voor een netwerk, bestaande uit twee schakelende werkstations en met transporttijden tussen de werkstations, is een heuristiek gepresenteerd om een goed bedieningsschema te verkrijgen. Allereerst wordt het netwerk als een enkel werkstation beschouwd en wordt het optimale periodieke schema bepaald. Vervolgens is dit schema geïmplementeerd in het netwerk en wordt voor elk werkstation de optimale faseverschuiving bepaald. Geschakelde werkstations met stochastische aankomstprocessen zijn ook besproken. Hier benaderen we de gemiddelde wachtrijlengte voor systemen met stochastische aankomstprocessen en gebruiken deze approximaties om de bedieningsschemas via een nietlineair programmeringsprobleem af te leiden. Als laatste zijn optimale besturingsstrategieën afgeleid voor dynamische systemen in discrete tijd met een onzekere vraag. Gebruik makende van technieken uit de theorie van optimale robuuste besturing (*robust optimal control*) kan een strategie bepaald worden, zonder zich van te voren te beperken tot een klasse van strategieën. We laten ook zien dat met het gebruik van waarnemers voor detecteerbare- of waarneembare systemen onbekende toestanden gereconstrueerd kunnen worden.

Dit proefschrift kan dienen als een beginpunt voor toekomstig onderzoek over besturing en een waarnemerontwerp voor schakelende werkstations. De geïntroduceerde methoden voor het bepalen van periodiek- en transiënt gedrag en het waarnemerontwerp kan uitgebreid worden naar netwerken met stochastisch gedrag en kan onderzocht worden voor realistische besturingsproblemen.

Curriculum Vitae

Dirk van Zwieten was born on November 28th, 1986 in Venray, The Netherlands. In 2004 he finished ‘Jeroen Bosch College’ in ’s-Hertogenbosch and started the Mechanical Engineering educational program at the Eindhoven University of Technology (TU/e). He received the master’s degree in 2010 after writing his master’s thesis on the modeling and analysis of biological networks with discrete event models, with a strong focus on the glycolysis pathway.

Dirk started research on control and observer design for switching servers in a Ph.D. project at Eindhoven University of Technology (supported by the Netherlands Organization for Scientific Research) in the Manufacturing Networks group of professor Rooda and professor Adan. Results of the project have been written in this thesis and have been presented at several conferences. During the Ph.D. project, Dirk was involved in a number of educational activities, such as tutoring bachelor students and coaching students in their bachelor or master projects. On September 9, 2014, Dirk defends his PhD thesis at Eindhoven University of Technology.