Performance analysis of networks on chips

DOI:
10.6100/IR657033

Document status and date:
Published: 01/01/2010

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

Performance Analysis of Networks on Chips

THOMAS STIELTJES INSTITUTE
FOR MATHEMATICS

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Beekhuizen, Paul

Performance analysis of networks on chips / by Paul Beekhuizen
– Eindhoven : Technische Universiteit Eindhoven, 2009.
Proefschrift. – ISBN 978-90-386-2144-9
NUR 919
Subject headings : Queueing Theory / Networks on Chips
2010 Mathematics Subject Classification : 60K25, 68M20, 90B18

Printed by Eindhoven University of Technology Printservice.
Cover design by Paul Verspaget.

Performance Analysis of Networks on Chips

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 donderdag 4 februari 2010 om 16.00 uur

door

Paul Beekhuizen

geboren te Rotterdam

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ir. O.J. Boxma

Copromotor:
dr. J.A.C. Resing

Acknowledgements

There are many people who, in one way or another, made it possible for me to complete this thesis, and I would like to thank a few of them in particular.

First of all, I would like to thank my daily supervisor and co-promotor Jacques Resing. We worked together on most of the research in this thesis, and I enjoyed our cooperation very much. I have especially benefitted a lot from Jacques' continuous search for the simplest non-trivial model, which proved insightful many times.

Next, I would like to thank Dee Denteneer from Philips Research for his guidance and supervision during my period as a PhD and an MSc student. Dee has had a prominent influence on my research over the last five years and I doubt that, without his efforts, I would have pursued a PhD.

I also wish to thank Onno Boxma for his supervision. Onno somehow manages to find time for anyone who can benefit from his help and he provided me with valuable feedback on multiple occasions, for which I am very grateful.

I am very grateful as well for the funding I received from Philips Research and for the freedom that was given to me. Thanks go out to my fellow PhD students at Philips for making my stay there a pleasant one.

During my period as a PhD student, I had an additional affiliation with EURANDOM, and I would like to thank everyone at EURANDOM for the great time I had there. EURANDOM has been a very lively and socially active environment, and working in such an environment seems almost indispensable for the successful completion of a PhD thesis. I will greatly miss the countless diners, poker nights, football and foosball matches, coffee breaks, social excursions, EURANDOM lunches, and all other events which I cannot think of at the moment.

As there cannot be a PhD defense without a defense committee a word of thanks is due to its president and its members: Ivo Adan, Harm Dorren, Rob van der Mei, Sindo Núñez Queija, and my three supervisors. Ivo is also one of the co-authors of [13] and I wish to thank him for our cooperation on that paper as well.

On a more personal note, I owe thanks to (all) my parents, the rest of my family, and my friends for their support and interest. Finally, I thank my wife Ivonne for her unconditional trust in my abilities and for supporting me in whatever I choose to do.

Contents

Chapter 1

Introduction

Modules on a chip, such as processors and memories, are traditionally connected via a single shared link (a bus). As chips become more and more complex, and the number of modules on a chip increases, this bus architecture becomes less efficient because a bus cannot be used by multiple modules simultaneously. Networks on chips are an emerging paradigm for the connection of on-chip modules.

In networks on chips, data is transmitted by packet switches, so that multiple links can be used at the same time and communication becomes more efficient. These switches have buffers, which leads to many performance-analytic questions that are interesting from both a theoretical and practical point of view. For example, one is typically interested in how much data can be transmitted by the network (throughput), how long it takes data to be transmitted (delay), how large buffers have to be to deliver a certain quality of service, and so on.

Due to the complex and unpredictable nature of data traffic, stochastic modelling and queueing theory play a key role in answering questions of this sort. In this thesis, stochastic models are developed and analysed in order to answer such questions and better predict and understand the performance of networks on chips.

In this introductory chapter, the characteristics of networks on chips are described in more detail. Furthermore, the key mathematical models are discussed, as well as relevant literature.

1.1 Networks on chips

Networks on chips have been proposed as a solution for the inefficiency caused by traditional bus connections in chips [20, 49]. In networks on chips, intellectual property blocks (*IP-blocks*, a general term for on-chip modules) are not connected to a single shared link, but to *network interfaces*. These network interfaces implement communication protocols, including tasks related to flow and congestion control, scheduling, routing, and so on.

Networks on chips use *switches* to transmit data across the network. A switch consists of input and output ports. Data packets arrive to the input ports of the switch, and leave from the output ports. If multiple input ports have data for the same output port, only one input port can transmit its data, and the switch selects which one. Data that is not transmitted immediately is stored in buffers and will be transmitted later.

Network interfaces are connected to IP blocks and switches by bidirectional *links*. Data transmissions over multiple links occur simultaneously because networks on chips are usually synchronised using a clock. This clock divides time into equal parts (*slots*), which entails that networks on chips operate in *slotted*, or *discrete* time. A schematic representation of a traditional chip and a network on chip can be found in Figures 1.1 and 1.2 respectively.

Besides higher efficiency, there are other practical advantages to networks on chips. One of them is a phenomenon called 'decoupling of computation and communication' [63,71,120]. Because communication protocols are implemented by network interfaces rather than IP-blocks, the computation and communication parts of the chip are separated: The IP-blocks take care of computations while the network takes care of communication. This decoupling entails that IP-blocks and network interfaces can be designed separately [114] and reused in multiple networks [93]. Another practical advantage is that networks on chips are reliable and energy efficient [19,20].

Queueing theory deals with the analysis of congestion phenomena caused by competition for service facilities with scarce resources. Such congestion phenomena

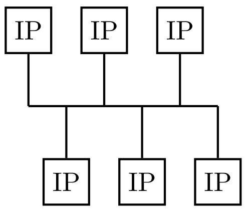


Figure 1.1: A bus architecture

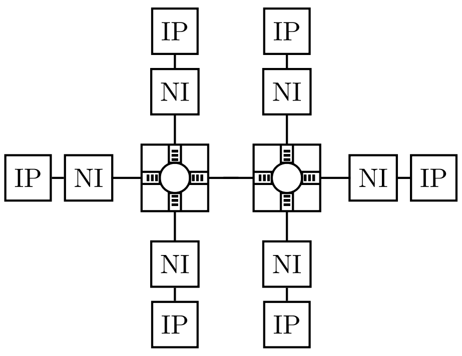


Figure 1.2: A network on chip

occur, for example, in computer networks, manufacturing systems, traffic intersections, and so on. These phenomena are typically analysed using stochastic models, which capture the unpredictable and uncertain nature of the processes giving rise to congestion (such as irregular arrival patterns of cars to an intersection).

In this thesis, we develop and analyse (stochastic) queueing models aimed at networks on chips. Due to the complexity and unpredictability of data traffic, stochastic models are useful tools for the performance analysis of these networks. For example, at present, performance validation of networks on chips is typically done using time-consuming simulation, which is not desirable in an optimisation loop. Using analytic models instead of or in addition to simulation can significantly speed up the design process [90, 108, 141].

There is a large amount of freedom in the actual implementation of networks on chips. As a result, many different implementations have been proposed, such as SPIN [5], Xpipes [21], Nostrum [102], and Hermes [103], and each proposed implementation has its own characteristics (see, e.g., [103] for a more comprehensive overview). This work is primarily motivated by Aethereal [64], the network on chip of NXP (formerly part of Philips).

In the remainder of this section, we describe the key features of networks on chips in general and Aethereal in particular. In Section 1.2, we discuss which types of packet switches exist, we describe their advantages and disadvantages, and we explain which type is used in networks on chips. We give a brief introduction to queueing theory in Section 1.3. Because networks on chips operate in discrete time, we consider discrete-time queueing models in this thesis. In such models, arrivals and departures occur at slot boundaries and the order in which they occur has important consequences. This is discussed in Section 1.3 as well. In Section 1.4 we describe the key models of this thesis. We review the structure of the thesis and mention the most important results in Section 1.5.

1.1.1 Quality of service

Two classes of traffic with a different quality of service are considered for networks on chips, namely *Guaranteed Services* (GS) and *Best Effort* (BE). With GS-traffic, a minimal throughput and a maximal delay are guaranteed and GS-traffic is therefore suitable for real-time communication. With BE-traffic such guarantees are not given. When data enters the network, the network simply gives its ‘best effort’ to transmit that data to its destination, without any guarantees as to when that data will arrive. BE-traffic is therefore more suitable if real-time communication is not required.

GS- and BE-traffic have different ways of resolving *contention*. Contention occurs when multiple data packets are trying to use one link simultaneously. As each link can only be used by one packet at the same time, the contention has to be resolved, i.e., one of the packets has to be selected for transmission.

Guaranteed services are naturally obtained using *circuit switching*, where contention is resolved by setting up connections in advance. For example, in Aethereal, every switch has a scheduling matrix S that determines which output port is reserved for which input port over a period of T time slots, i.e., if $S(t, o) = i$ then output port o is reserved for input port i in time slot t (modulo T). If slot t is reserved for some data on a certain switch, then slot $t + 1$ (modulo T) must be reserved for that data on the next switch on its path, and so on [120].

Data transmission over a connection is deterministic because data from two different connections never interfere. In particular, this means that every connection receives a fixed throughput depending on how many reservations have been allocated to that connection. Furthermore, because data is transmitted over a reserved connection, no delay is incurred in the network, which explains why GS-traffic is suitable for high priority real-time communication.

The disadvantage of GS-traffic is that links have to be reserved for worst-case scenarios. After all, to guarantee sufficient resources for a connection, the allocation of links has to be based on the *maximal* required throughput of that connection. However, if the *actual* required throughput is (temporarily) lower, part of the reserved links remain unused, which results in poor link utilisation [63, 120].

BE-traffic uses *packet switching* to transmit data across the network. With packet switching, contention is resolved by the switches: Data is divided into packets by the network interfaces, and a header with routing information is added. The packets are then transmitted to the switch without any a priori scheduling. When the packet arrives to a switch, that switch decides which packets to transmit, based also on other packets present.

The behaviour of BE-traffic thus depends on other packets, which makes it more stochastic in nature than GS-traffic, and BE-traffic is therefore typically lower priority traffic for which real-time communication is not required. Although the average performance of BE-traffic is better than that of GS-traffic because unused reserved links are not lost, the downside of BE-traffic is that it is unpredictable due to its stochastic nature [120].

GS-traffic is easy to analyse and predict using deterministic models. For the analysis of BE-traffic, however, stochastic models play a key role. In this thesis, we therefore focus on queueing models specifically aimed at BE-traffic.

1.1.2 Flow control

Due to the stochastic nature of BE-traffic, it is in principle possible that packets arrive at a full buffer, resulting in packet loss. Networks on chips therefore implement two types of flow control regulating traffic across the network: Link-to-link flow control, which regulates traffic between switches, and end-to-end flow control, which regulates traffic between network interfaces.

There are three forms of link-to-link flow control [103]: *store-and-forward, virtual cut-through,* and *wormhole routing.* With *store-and-forward*, an entire packet is stored in a queue of the switch before it is sent to the next switch. This requires enough buffer space for the entire packet at each switch, and the packet is delayed at each switch until the packet has arrived entirely. This type of flow control hence requires large buffers and has a large delay.

With *virtual cut-through*, a packet is forwarded as soon as the next switch has enough buffer space available to store the entire packet. It is thus faster than store-and-forward, but it still requires large buffers.

With *wormhole routing*, packets are divided into *flits*, where a flit is the amount of data that can be transmitted over a link in one time slot. With wormhole routing, a flit is forwarded to the next switch if it has space to store one flit. Once the first flit of the packet has been sent via a certain output port, that output port remains reserved for all flits of that packet. One packet may thus be spread over multiple switches. Because flits are stored instead of entire packets, wormhole routing requires the least buffer space. Since buffer space is expensive, most networks on chips, including Aethereal, use wormhole routing [103].

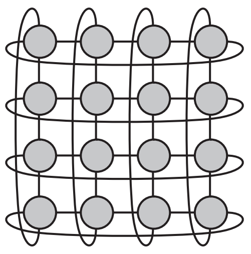
In addition to link-to-link flow control, networks on chips also implement end-to-end flow control, which regulates traffic between source and destination network interfaces. In Aethereal, for example, credit-based flow control is implemented [114]. With this form of flow control, the number of flits from one network interface to another is restricted to a maximum. When the destination network interface forwards data to the IP-block connected to it, an acknowledgement is sent back in the form of credits indicating how much additional data the source network interface may send. These credits are either included in data from the destination back to the source (piggybacked) or sent by themselves.

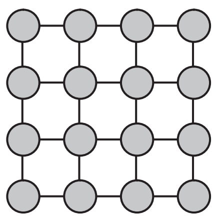
Throughout this thesis, a key assumption is that switches indeed use wormhole routing. Furthermore, we study a specific class of networks operating under end-to-end flow control (see Section 1.4 and Chapter 7).

1.1.3 Network topologies

The physical positioning of switches in a network is called the topology. Many different topologies are considered for networks on chips, such as a mesh, a torus, a tree, or a ring-based topology, or mixtures of such topologies (see also Figure 1.3). Every topology has its own advantages and disadvantages and different network on chip proposals use different topologies. For an overview of topologies used, we refer to [25] and [103].

Although different topologies are considered, most networks on chips, including


(a) Torus


(b) Mesh

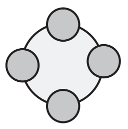

(c) Ring

Figure 1.3: Three different topologies for switches in networks on chips [25]: Torus, mesh, and ring.

Aethereal, use the mesh topology [103]. With this topology, switches are placed on a lattice with connections in four directions (up, down, left, right). Bjerregaard and Mahadevan [25] further distinguish indirect and direct networks. With direct mesh networks, every switch is connected to a network interface. The number of input and output ports per switch thus ranges from 3 (in the corners) to 5 (in the center). With indirect mesh networks, some switches are connected to network interfaces but others are not. An example of the latter is a network where network interfaces are only connected to the switches on the edges, in which case all switches have 4 input and output ports. The difference between direct and indirect mesh networks is illustrated in Figure 1.4.

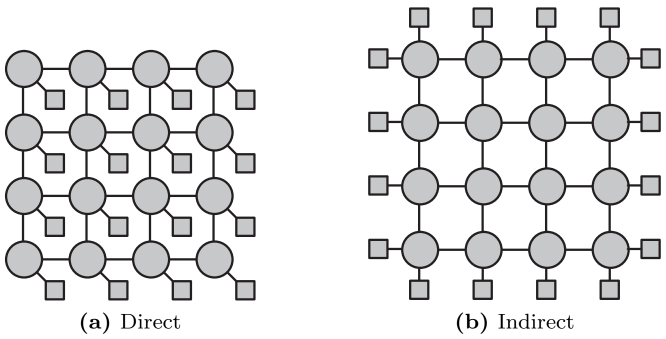


Figure 1.4: Direct and indirect mesh networks.

Closely related to the concept of topologies is the concept of a routing discipline. A routing discipline dictates which route traffic from one IP-block to another takes. Routing disciplines can be either *deterministic* or *adaptive.* With deterministic routing disciplines, the route of traffic from one IP-block to another is always the same. With adaptive routing, the routes may differ based on the amount of traffic in the network.

A popular routing discipline for mesh topologies in networks on chips is the XY-routing discipline. With this deterministic routing discipline, traffic always first traverses the network horizontally, as far as it has to go, and then vertically, towards its destination. XY-routing is used in many networks on chips due to its simplicity.

Motivated by mesh networks, we consider switches with only a small number of ports, say 4 or 5. Furthermore, we assume that XY-routing is used. In particular, this ensures that specific mesh networks fall into the class of concentrating tree networks, which is one of the key models studied in this thesis (see Section 1.4.2).

1.2 Switches

A switch is a device that transmits data packets from one link to another. Packets arrive over links connected to *input* ports of the switch, and leave over links connected to *output* ports of the switch. If multiple packets have the same destination, only one of them can be transmitted and the switch has to select which one. Packets that cannot be transmitted have to be buffered and they will try to reach their destination again in the next time slot. A schematic representation of a switch can be found in Figure 1.5.

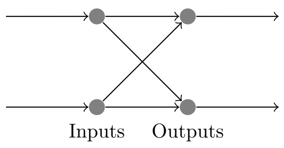


Figure 1.5: An abstract representation of a switch.

Packet switches have been studied extensively as part of communication networks such as the internet, local area networks, and ATM networks, and there is a variety of different switch architectures with different methods to provide buffering. In this section, we discuss which buffering strategies exist, we give an overview of their performance, and we explain which type is used in networks on chips.

In the remainder of this thesis, we say that N is the number of input ports of a switch and M the number of output ports. A switch with N input ports and M output ports is called an $N \times M$ switch. Furthermore, to simplify the description of the different switches, we assume in this section that all packets consist of one flit, which means that a packet requires precisely one time slot to be transmitted.

1.2.1 Buffering strategies

In this subsection, we discuss four different buffering strategies for switches, namely combined input output queueing, output queueing, input queueing, and virtual output queueing.

Combined input output queueing

The most general switch architecture considered in this section is a *combined input output queueing* (cioq) switch. Cioq-switches have a *speedup* s; each time slot is divided into s phases, with s between 1 and N. In each phase packets are switched from inputs to outputs, with the restriction that in each phase only one packet may be switched from an input port, and only one packet may be switched to an output port, i.e., each input and output port may be used only once per phase. Up to s packets are thus switched per port per time slot, whereas each time slot only one packet can be transmitted over a link, so the switch operates s times as fast as the links connected to it.

In each phase, the switch uses a scheduling algorithm to decide which input may transmit to which output. A common way to do so is by finding a maximum weight matching in a bipartite graph. One vertex set of this graph is given by the inputs, and the other by the outputs. There is an edge between the input i vertex and the output j vertex if the first packet (the *Head-of-Line* packet, or *HoL-packet*) in input queue i has destination j. The weights given to an edge between the input i and output j vertices can, for instance, be equal to the length of input queue i, which leads to the longest queue first discipline, or to the waiting time of the HoL-packet of queue i, which leads to the oldest packet first discipline, etc.

As a result of the speedup, cioq-switches must have queues on the inputs and outputs to prevent packet loss: Even though up to s packets can be switched to their output ports, the links connected to the output port may transmit only one packet per time slot. The switch must thus have buffers at the *outputs*. Likewise, up to N packets with the same destination may arrive at all input ports together per time slot, but only s of them can be actually switched to their destination. The switch must thus have buffers at the *inputs* as well.

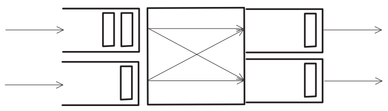


Figure 1.6: Combined input output queueing

Output queueing

A notable special case of a cioq-switch is a cioq-switch with a speedup of N. In this case, up to N packets may be switched to the same output port per time slot. Because at most N packets with the same destination arrive per time slot at all input ports combined, all arriving packets can be switched. This implies that only queues on the outputs are needed, hence the name *output queueing*.

Another variant of an output queued switch is a switch where each output port is equipped with N separate queues; one for each input, i.e., packets from input i to output j are stored in queue i of output j. With this strategy, a speedup is not needed; the switch operates at speed 1.

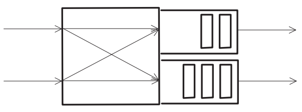


Figure 1.7: Output queueing

Input queueing

Another special case of a cioq-switch is a switch with a speedup of 1. In this case, at most one packet is switched to each output per time slot. Output buffers are thus not needed, and the switch is called an input queued switch.

As will be explained in Section 1.2.3, input-queued switches are commonly used in networks on chips. Input-queued switches are therefore the most important switches of this thesis.

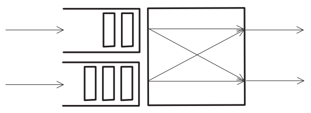


Figure 1.8: Input queueing

Virtual output queueing

Cioq-switches can be combined with a strategy called *virtual output queueing* (voq). With virtual output queueing, all N input queues are subdivided into M separate queues such that every queue only stores packets with the same origin and destination, as displayed in Figure 1.9.

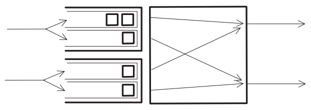


Figure 1.9: Virtual output queueing

In practice it is not always necessary to actually use M *physically* separate queues; it is also possible to still use one buffer per input. In this case, however, the order in which packets depart is no longer First-In-First-Out (FIFO), but one that depends on the destinations of packets in the other queues. Virtual output queueing is thus mainly a change in the order in which packets depart from a queue; instead of FIFO, a more dynamic and complicated order is used.

In most applications, input-queued switches are combined with virtual output queueing. In fact, in literature the term input-queued switch often refers to a switch with queues at the inputs, regardless of whether it is combined with virtual output queueing or not. To emphasise the difference between input-queued switches with a *single* FIFO queue per input and input-queued switches with virtual output queueing, we will refer to the former as *single input queueing* (siq).

1.2.2 Throughput

Perhaps the most important performance characteristic of a packet switch is its throughput. The throughput of an input port is defined as the mean number of packets transmitted from that port per time slot. Because at most one packet arrives per time slot at each input port, an important property of a switch is whether it has a throughput of 1, which is the case if *all* its input ports can sustain a throughput of 1. If the switch indeed has a throughput of 1, it has enough capacity to transmit all incoming traffic. If it cannot, packet loss may occur if the load is too high.

In this section, we briefly overview relevant throughput results. For a more elaborate literature review on switches the reader is referred to [151].

Karol et al. [77] studied *uniform* $N \times N$ single input queued switches. Uniform means that the arrival rates to all input ports are the same, packet destinations are given by i.i.d. random variables, and every destination has probability $1/N$ of occurring. Karol et al. showed that these switches suffer from *Head of Line blocking*, or *HoL-blocking*. HoL-blocking occurs when the packet in the first position of a queue cannot be transmitted because another packet has the same destination, while the destination of the packet in the second position is available. As a result of HoL-blocking, the throughput of a uniform siq-switch is limited to $2 - \sqrt{2} \approx 0.586$ if the number of ports tends to infinity and all buffers are infinitely large.

Karol et al. also studied uniform $N \times N$ output-queued switches (without the assumption that N tends to infinity), and argued that such switches have a throughput of 1. The disadvantage of output-queued switches, however, is that they cannot always be used in practice due to the speedup of N.

Due to the poor performance of single input queued switches, most research aimed at improving that performance. For instance, Karol et al. [76] considered a switch where, instead of only packets in HoL-positions, the first few packets may be transmitted, which improves throughput. Kolias and Kleinrock [85] suggested dividing each input queue into 2 separate queues such that all packets with an even destination arriving at a particular input are stored in the even queue of that input, and all packets with an odd destination in the odd queue of that input. They later extended this to m queues per input [86] and coined the term virtual output queueing for $m = M$, although the principle itself had already been introduced before in [136].

The performance of switches with virtual output queueing depends on the precise scheduling algorithm used, so scheduling algorithms are an important research topic, see e.g. [98–100, 125]. In particular, it has been shown that a throughput of 1 can be achieved with the longest queue first and oldest packet first disciplines [46, 101].

Besides improving the performance of a single input queued switch, it is also possible to reduce the speedup of output queued switches (which are essentially cioq-switches with a speedup of N) to make them more practically feasible. As discussed, this comes at the cost of having to introduce buffers at the inputs. Combined input output queued switches with a speedup lower than N have been studied extensively, for example in [39, 70, 73, 79, 109, 111].

It has been shown that cioq-switches do not need a speedup of N to achieve a throughput of 1. In fact, for cioq-switches with virtual output queueing, *any* non-idling scheduling algorithm obtains a throughput of 1 if the speedup is equal to two [46]. Non-idling means that no packet has to wait unnecessarily, i.e., if a packet at input i has destination j and it is not scheduled, another packet from input i must have been scheduled, or a packet from another input must have been scheduled to output j. Moreover, a cioq-switch with a speedup lower than N can mimic a cioq-switch with a speedup of N (i.e., an output-queued switch) exactly [89, 113], even without virtual output queueing [40]. Mimicking means that two switches with sample-path wise identical arrival processes have sample-path wise identical departure processes.

1.2.3 Switches in networks on chips

In networks on chips, the physical area of switches is the dominant factor in the costs of the network [62]. Output queued and combined input output queued switches require many buffers and are therefore too expensive [69, 120]. Virtual output queueing can be implemented with a single buffer but then it requires the use of RAM (Random Access Memory) instead of FIFO queues [120]. For our purposes, the main difference between RAM and FIFO is that in RAM, packets can be removed from *any* position in the memory, whereas in a FIFO queue only the *first* packet can be removed. RAMs generally occupy a large area [64,66,146], which makes them expensive as well.

Siq-switches have only few queues, and the queues they do have are cheap FIFO queues [146]. Siq-switches are thus much cheaper than the more advanced types of switches. Although some networks use the more advanced switches, siq-switches are used in most networks on chips proposed in literature [103], despite their poorer performance.

In Aethereal, scheduling in siq-switches is performed using a round robin scheduler [120]. With such a scheduler, every output port j has an index c_j referring to an input queue. The input queues are considered for transmission in cyclic order starting at input queue c_j, and the first input queue that has a HoL-packet with destination j can transmit that packet: First, input queue c_j is considered, and if it has a HoL-packet with destination j that packet is transmitted. Second, input queue $c_j + 1 \mod N$ is considered and if its HoL-packet has destination j that packet is transmitted, and so on.

After switching a packet from an input port, the value of c_j changes; if a packet from queue i was transmitted to output j, c_j is set to the value $i + 1 \mod N$. If no packets were transmitted through output j the value of c_j remains the same.

We focus on the performance analysis of siq-switches in this thesis. Furthermore, because siq-switches are the only switches we consider, we will simply refer to them as 'switches' from now on.

1.3 Queueing theory

In this section, we give a brief introduction to queueing theory. In Section 1.3.1, we discuss queueing models in general. Because networks on chips are synchronised using a clock, packet transmissions over multiple links occur simultaneously, which gives rise to discrete-time queueing models. In such models, arrivals and departures occur at slot boundaries and the order in which they occur is important. This is discussed in Section 1.3.2.

1.3.1 General queueing systems

The most elementary queueing model deals with the situation where jobs arrive to a single service facility called the server. Jobs are served by the server and leave the system when their service has been completed. If a job arrives when another job is in service, the arriving job is placed in a queue, also called a 'buffer'. When the server completes service of a job, it starts service of one of the jobs from the queue, and so on. A schematic representation of this situation can be found in Figure 1.10.

The model described above is an abstraction of many real-life situations that involve queueing. One example is that of a supermarket, where customers with shopping carts have to pass the checkout; the jobs are the customers with shopping items and the server is the cashier. Another example is that of a call centre, where the jobs are phone calls by customers, and the server is an operator handling these calls. In communication networks, the jobs are data packets and the server can be a switch or a link.

Using this queueing model, the performance and effectiveness of systems can be assessed. For example, in communication networks, one is typically interested in the time spent by packets in the buffer (the *waiting time*), the number of packets served per time unit (the *throughput*) and the number of packets waiting in the buffer (the *queue length*).

Job arrivals to a queueing system are typically unpredictable. To model this unpredictability, it is commonly assumed that jobs arrive according to a stochastic process, such as a Poisson process. Likewise, the service time (the time a job spends in service) is often unpredictable, and therefore assumed to be stochastic as well. Besides the arrival process and service time distribution, the number of available buffer positions and the number of servers can also be varied.

Kendall introduced a four-symbol notation $A/B/C/D$ that is used to describe queueing systems. For the first symbol, A, a letter is substituted that describes the distribution of the interarrival time, the time between two consecutive arrivals. Examples are M for the exponential (memoryless) distribution leading to a Poisson arrival process, D for deterministic interarrival times leading to a periodic arrival process, and G for generally distributed interarrival times. For the second symbol, a letter describing the distribution of the service times is substituted. Again, M stands for exponential, D for deterministic, G for general, and so on. For the third symbol, C, the number of servers in the queueing system is substituted. Finally, for the fourth symbol, D, the number of available buffer positions is substituted. If the buffer is infinitely large, the last symbol is usually left out. For example, an $M/G/1$ queue is a queue with a Poisson arrival process, generally distributed service times, a single server, and infinitely many buffer positions.

One of the most elementary results from queueing theory is Little's law (see, e.g., [130, 145]). Little's law relates the mean queue length to the mean sojourn time (defined as the mean waiting time plus the mean service time): $\mathbb{E}[Y] = \lambda\,\mathbb{E}[S]$,

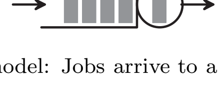


Figure 1.10: A basic queueing model: Jobs arrive to a server and are placed in a queue if they cannot be served immediately.

where Y is the queue length, including one job in service if there is one, λ is the arrival rate, i.e., the expected number of jobs arriving per time unit, and S is the sojourn time. Another important tool in the analysis of queueing systems is PASTA (Poisson Arrivals See Time Averages, see [148]). The PASTA property states that, with a Poisson arrival process, the number of jobs in the buffer at an arbitrary point in time is in distribution equal to the number of jobs in the buffer immediately before the arrival of another job. The PASTA-property is, for example, very useful for the analysis of waiting times in the $M/G/1$ queue.

Many variants of this basic model have been studied. For example, most queueing systems employ the FIFO service order, but other orders, such as LIFO (Last-In-First-Out), SIRO (Service-In-Random-Order), and processor sharing, where all jobs receive a fraction of the capacity of the server, have also been considered. A more complex variant is a model where multiple queues share a single server. Finally, we mention the possibility that jobs leaving a server are sent to another server, leading to *networks* of queueing systems.

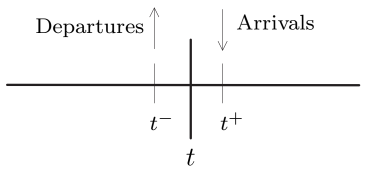
For a more elaborate introduction to queueing theory, the reader is referred to the introductory books by Kleinrock [82, 83] or the lecture notes by Adan and Resing [3]. Due to our focus on queueing models for networks on chips we consider packets rather than jobs arriving to a server in the sequel.

1.3.2 Arrival models in discrete-time queueing systems

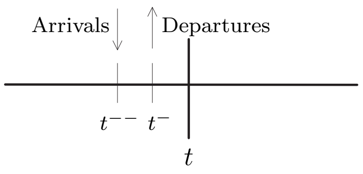
In networks on chips, packet transmissions over all links are synchronised using a clock, which effectively means that networks on chips operate in discrete, or 'slotted' time. It is therefore natural to consider *discrete*-time queueing models for networks on chips. *Continuous*-time queueing models, however, are much more popular in queueing theory literature. The main difference between continuous- and discrete-time models is that in discrete time, arrivals and service completions (departures) occur simultaneously at slot boundaries; something which happens on a continuous time scale with probability 0.

Although arrivals and departures occur simultaneously, one has to specify an order between them for the sake of analysis. The choice of this order (called arrival model) has consequences for the applicability of Little's law and BASTA (Bernoulli Arrivals See Time Averages, see [32]), the discrete-time equivalent of PASTA. Other than such fundamental issues, the choice of the right arrival model turns out to be important when networks of queues are considered. In this section, we therefore discuss the effects of different arrival models.

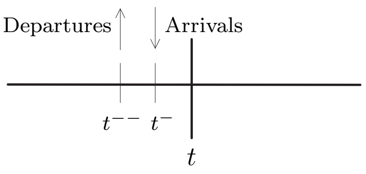
We consider three arrival models studied by Desert and Daduna [50]: The early arrival (ea) model, the late arrival - arrivals first (la-af) model, and the late arrival - departures first (la-df) model. Desert and Daduna describe the different arrival models by introducing time epochs $t^{--} < t^{-} < t < t^{+}$, for any slot boundary $t \in \mathbb{N}$, where the difference between each is infinitesimal. In the ea-model, arrivals take place at the beginning of time slots, i.e., at t^{+}, and departures at the end, i.e., at t^{-}. A packet arriving at time t^{+} may be served in time slot $[t, t+1)$. In the late arrival models, arrivals and departures occur at the end of time slots, with either



(a) Early arrival



(b) Late arrival - arrivals first



(c) Late arrival - departures first

Figure 1.11: Three different arrival models, early arrival, late arrival - arrivals first, and late arrival - departures first.

arrivals before departures (la-af), i.e., arrivals at t^{--} and departures at t^-, or the other way around (la-df). The three arrival models are depicted in Figure 1.11.

We denote the number of packets seen by an arbitrary arriving packet by L, and the number of packets at an arbitrary slot boundary t by Q. The arrival model is added as a subscript: L_e, L_a, and L_d are the number of packets seen in the ea-model, the la-af-model, and the la-df-model respectively, and likewise for Q. We furthermore denote the number of packets at an arbitrary point in the continuous time domain by Y. Because the arrival models only change the behaviour at infinitely small time intervals, Y is the same for all arrival models (unless arrivals or service completions are state-dependent, see [50]).[†]

Bernoulli Arrivals See Time Averages

We consider the most general single-server queue with a Bernoulli arrival process, namely a $Geo/G/1$ queue. In this queue, packet arrivals take place every time slot with a fixed probability (so the interarrival times are geometrically distributed), and every packet has a generally distributed service time. The BASTA property states that the queue length seen by an arriving packet is equal to the queue length at arbitrary times. There are, however, two different interpretations of the queue length at arbitrary times, namely that at an arbitrary point in the *continuous*-time domain (see, e.g., [67]) and the *discrete*-time domain (see, e.g., [50]).

Gravey and Hébuterne [67] show that BASTA holds with respect to *continuous*-time queue lengths if and only if arrivals occur before departures at slot boundaries. Since this is only the case for the la-af-model, we have, in our notation, $L_a \overset{d}{=} Y$,

[†]The quantity Y can also be viewed as the number of packets at times $t + 1/2$, with $t \in \mathbb{N}$. It is also sometimes called the number of packets seen by an outside observer.

$L_d \overset{d}{\neq} Y$, and $L_e \overset{d}{\neq} Y$. Here, $\overset{d}{=}$ denotes equality in distribution. Note that we used that, in the ea-model, departures indeed occur before arrivals at slot *boundaries*, even though arrivals occur before departures *within* a time slot.

For BASTA with respect to *discrete*-time queue lengths, it is easily shown by a sample-path argument that the queue length at slot boundaries is in distribution equal to the queue length at an arbitrary point in continuous time for the late arrival models: $Q_a \overset{d}{=} Y$ and $Q_d \overset{d}{=} Y$. It thus follows that $L_a \overset{d}{=} Q_a$ and $L_d \overset{d}{\neq} Q_d$. For the early arrival model, we refer to Takagi [134], where it is shown that $L_e \overset{d}{=} Q_e$.

Little's law

Care is thus in order with the BASTA property. Care is also in order with the application of another fundamental result in queueing theory: Little's law. For any arrival model, Little's law relates the mean sojourn time to the mean queue length in the *continuous-time* domain [130], rather than that at discrete times. In other words, Little's law states $\mathbb{E}\, Y = \lambda\, \mathbb{E}\, S$, where λ is the arrival rate and S the sojourn time. It follows that Little's law only relates the mean sojourn time to the mean queue length at *discrete* times if the mean queue length at discrete and continuous times are the same. In general, this only holds for the late arrival models, because $Q_a \overset{d}{=} Q_d \overset{d}{=} Y$ and $Q_e \overset{d}{\neq} Y$.

That Little's law cannot be applied thoughtlessly to discrete-time queue lengths is especially apparent in the $D/D/1$ queue with unit interarrival and service times: With the early arrival model, every time slot a packet arrives and departs in that same time slot, i.e., for any time $t \in \mathbb{N}$ a packet arrives at time t^+ that leaves at $(t+1)^-$. The system is thus always empty at discrete times, but always non-empty in the continuous time domain; $Q_e = 0$ and $Y = 1$. Applying Little's law to Q_e instead of Y would lead to the rather odd conclusion that the sojourn time is equal to 0, even though the load is 1 and every packet spends precisely one time slot in the system.

Networks of queues

Desert and Daduna [50] also analyse the effects of different arrival models when queues are put in a tandem network. For the la-af-model, a packet served at time t^- arrives at the next queue at time $(t+1)^{--}$, which means the packet disappears from the network for one time slot. To prevent such irregularities, one could, with 2 queues in an ordinary tandem, introduce additional time epochs t^{----} and t^{---} such that arrivals at queue 1 occur at t^{----}, packet departures from queue 1 at t^{---}, packet arrivals at queue 2 at t^{--} and packet departures from queue 2 at time t^-. With J queues in tandem a similar solution with $2J$ time epochs is possible but for more general topologies, a similar solution may become very cumbersome. The la-af-model is thus not a natural choice for networks of queues.

For the la-df model, any packet served at t^{--} arrives at the next queue at time t^-. Likewise, for the ea-model any packet that is served at time t^- arrives at the

Model	Relations	$\mathbb{E}\,Q = \lambda\,\mathbb{E}\,S$?	Networks?
Early arrival	$L_e \overset{d}{=} Q_e \overset{d}{\neq} Y$	No	Yes
Late arrival - arrivals first	$L_a \overset{d}{=} Q_a \overset{d}{=} Y$	Yes	No
Late arrival - departures first	$L_d \overset{d}{\neq} Q_d \overset{d}{=} Y$	Yes	Yes

Table 1.1: The differences between various arrival models. Here L is the queue length seen by an arriving packet, Q that at discrete-time epochs $t \in \mathbb{N}$, and Y that at an arbitrary point in the continuous-time domain.

next queue at time t^+. The ea- and la-df-models are thus more natural models for networks, because peculiarities like in the la-af case do not occur.

Summary

The differences between the various arrival models are summarised in Table 1.1. All three arrival models have their peculiarities: For the la-df-model, BASTA does not hold, for the ea-model, Little's law only holds for the continuous-time queue length, and the la-af model is not very suitable for networks.

In this thesis, suitability for networks is important, so we do not consider the la-af-model. The ea-model and the la-df-model differ only in the point at which queue lengths are observed; either between or after departures and arrivals. Apart from this difference the queue lengths in both models are sample-path wise the same. We can therefore assume an arrival model based on the properties we want the queueing system to have. Applicability of Little's law will turn out to be useful, so we assume the la-df arrival model throughout this thesis.

1.4 Models

The research of this thesis is centred around two key models: The first one is a model of only one switch, a so-called single-switch model. The second is a network of polling stations, which is motivated by a network on chip where all traffic has the same destination. Both models are described in more detail in this section.

1.4.1 Single-switch models

Consider a model of an $N \times M$ switch as depicted in Figure 1.12. We assume that packets arrive at queue i of the switch according to a Bernoulli process with parameter λ_i (i.e., every time slot an arrival takes place with probability λ_i, independently of all previous time slots). A packet arriving to queue i has output j as destination with probability p_{ij}, independently of everything else. Recall that if $\lambda_i = \lambda$ and $p_{ij} = 1/M$, the switch is called uniform.

The switch constitutes a discrete-time process $(D(t), Q(t))$, where the vector $D(t) = (D_1(t), \ldots, D_N(t))$, and $Q(t) = (Q_1(t), \ldots, Q_N(t))$. Here, $D_i(t)$ denotes the destination of the HoL-packet in queue i at time t, and $Q_i(t)$ denotes the length of

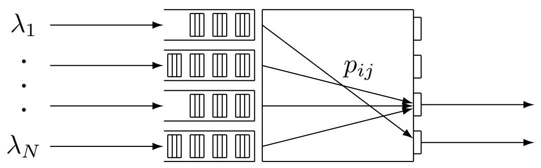


Figure 1.12: A schematic representation of the model of a switch. Packets arrive at rate λ_i to queue i of the switch, and have destination j with probability p_{ij}.

queue i at time t. If queue i is empty, we say $D_i(t) = 0$.

If the switch uses the random order discipline (i.e., if there are k HoL-packets with the same destination, each of them is selected with probability $1/k$), the process $(D(t), Q(t))$ is a Markov chain: The arrivals and destinations of packets are given by independent random variables and departure probabilities can be derived from $D(t)$ due to the random order discipline. In the remainder of this subsection, we discuss possibilities to analyse this Markov chain.

Saturated switches

Switches are sometimes studied in a state known as *saturation*. Saturation is an overload situation where all queues always have packets, i.e., $Q_i(t) \geq 1$ for all i and t. Every transmitted HoL-packet is thus immediately replaced by a new packet and the process $D(t)$ itself constitutes a Markov chain.

The state space of this Markov chain is $\{1, \ldots, M\}^N$, and the Markov chain is in state $x = (x_1, \ldots, x_N)$ if the HoL-packet at queue i has destination x_i. The transitions of this Markov chain are caused only by departures of old packets and arrivals of new HoL-packets with new destinations. The number of packets with the same destination, and hence the departure probability, can be derived from x. The new HoL-packets have destinations that are independent of everything else.

Because the process $D(t)$ is a Markov chain on a finite state space, its equilibrium distribution can be found numerically using straightforward techniques. Moreover, using this equilibrium distribution, important throughput results can be derived.

Remark 1.4.1. For general switches in saturation, the process $D(t)$ is indeed a Markov chain on the state space $\{1, \ldots, M\}^N$. For *specific* switches such as a uniform switch, simplifications of the state space are possible. Using these simplifications the size of the state space and the computational burden can sometimes be reduced significantly. See, e.g., [12, 30] for details.

Non-saturated switches with finitely many input queues

We consider again a non-saturated switch described by the discrete-time process $(D(t), Q(t))$. With infinite buffers, the state space of the process $(D(t), Q(t))$ consists of a finite number of parallel N-dimensional planes $\{0, 1, \ldots\}^N$; for every possible destination vector $D(t)$ (of which there are finitely many), all N queue lengths take a value in $\{0, 1, \ldots, \}$.

The process $(D(t), Q(t))$ is a *spatially homogeneous* Markov chain. Spatially homogeneous means that transitions in the interior of a plane happen with the same probability, regardless of the precise position on that plane, and likewise for the boundaries. In other words, packet arrivals and departures may only depend on whether or not queues are empty, and not on the number of packets in the queues.

Spatially homogeneous Markov chains on *single* planes in *two* dimensions have been studied extensively (for a textbook see [54]), for example in the context of cable networks [118, 138, 139], coupled processors [53], and so on. Nevertheless, exact analysis of spatially homogenous 2-dimensional Markov chains is very hard in general. In [2], three approaches for particular classes of such Markov chains are discussed: The compensation approach [1, 4], translation to a 2-dimensional boundary value problem from mathematical physics [44], and the uniformisation technique (see e.g. [56, 81]). These approaches have also been applied to 2×2 *output-queued* switches: The compensation approach was applied by Boxma and Van Houtum [35], the uniformisation approach by Jaffe [75], and the boundary value approach by Jaffe [74] and Cohen [41, 42].

If a 2×2 *single input-queued* switch has uniform destinations ($p_{ij} = 1/2$ for all i and j), it also gives rise to a Markov chain on a *single* 2-dimensional plane. Due to the uniformity of destinations, contention (both HoL-packets having the same destination) occurs with probability 1/2 if both queues are non-empty, regardless of the destinations in previous time slots. Additional parallel planes are thus not needed; the process $Q(t) = (Q_1(t), Q_2(t))$ itself is already Markovian.

With general values of p_{ij}, however, a 2×2 siq-switch already constitutes a Markov chain on a number of *parallel* 2-dimensional planes. After all, the probability of contention in a certain time slot depends on the destinations of HoL-packets in the previous time slot: If both HoL-packets have the same popular destination, only one new packet will move to the HoL-position and contention is very likely, but if there is contention for a less popular destination, contention is less likely in the next time slot. The destinations of packets must thus be taken into account to make the process Markovian, i.e., the process $Q(t)$ is not Markovian, but $(D(t), Q(t))$ is.

For a general $N \times M$ siq-switch, determining the queue length distribution requires solving a Markov chain on a finite number of parallel planes in N dimensions. Yet, even Markov chains on a *single* N-dimensional plane with $N > 2$ have eluded researchers so far; a general approach to solve such Markov chains has not yet been discovered. Moreover, obtaining results for 2-dimensional models requires advanced techniques from complex function analysis, and the derivations of these results do not offer much hope for extensions to higher dimensions. Although the equilibrium distribution of a 2×2 siq-switch with uniform destinations can probably be obtained in exact form using one of the techniques discussed in [2], we focus on approximations for more general cases in this thesis, rather than pursuing an exact analysis for this one special case.

Non-saturated switches with infinitely many input queues

Karol et al. [77] analysed the mean queue length of a *uniform* $N \times N$-switch with Bernoulli arrivals under the assumption that N tends to infinity. Their analysis was based on two observations: First, if N tends to infinity, the lengths of the input queues become independent. Second, if N tends to infinity, the number of packets with the same destination arriving at HoL-positions follows a Poisson distribution. These two observations together imply that the time a packet spends in the HoL-position is equal to the sojourn time in a discrete-time $M/D/1$ queue with random order of service. In particular, this allows for analysis of the mean queue length.

Li [95] analysed the mean queue length and throughput of a *non-uniform* $N \times N$ switch under the assumption that $N \to \infty$, using the same two observations. Furthermore, he studied a switch with geometric packet sizes in [96]. For uniform switches the analysis of Li corresponds to that of Karol et al. [77].

If these asymptotic results are applied to switches with a finite number of queues, they only yield *approximations*. These approximations are generally accurate for large N. In mesh topologies, however, the size of switches is usually 4 or 5 (see Section 1.1.3) and the asymptotic analysis of uniform switches leads to quite inaccurate approximations for small N, as is shown in Chapter 2.

1.4.2 Concentrating tree networks of polling stations

The second key model considered in this thesis is a network of polling stations. A polling station is a queueing system where multiple queues are served by a single server. In the remainder of this subsection, we describe the network model in more detail, and we give a brief introduction to polling systems.

Network model

Consider a concentrating tree network of polling stations, as displayed in Figure 1.13. Packets arrive to the network from external sources and are served by the polling station to which they arrive. After a packet completes service at this station, it moves to another polling station, where it is served again, and so on. All packets in the network move towards a single node, called the sink. After service at the sink, the packets leave the network.

The concentrating tree network model is motivated by networks on chips where all traffic has the same destination, which happens for example if multiple masters (e.g., processors) share a single slave (e.g., memory). As described in Section 1.1.3, switches in networks on chips are typically organised in a mesh topology, and the predominant routing discipline is XY-routing. With this routing discipline, packets first travel across the network horizontally, and then vertically. An example of a mesh network with XY-routing where all traffic has the same destination is displayed in Figure 1.14.

As can be seen from Figure 1.14, the mesh network topology combined with XY-routing is a special case of a concentrating tree network. Furthermore, because

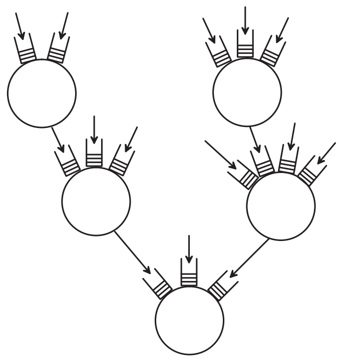


Figure 1.13: A concentrating tree network of polling stations.

all traffic has the same destination, every switch has several queues sharing a single link connecting that switch to the next. Every switch can thus be seen as a server attending multiple queues, i.e., as a *polling station*, and a *network* of switches as a network of polling stations.

In addition to *open* concentrating tree networks of polling stations, where packets arrive from the exterior, we also study a *closed* model for a concentrating tree network operating under flow control. In closed models, packets immediately reenter the network after service at the sink. Effectively, packets thus remain the network forever and number of packets in the network is fixed at all times. An alternative way of looking at this network is that the network starts with a certain number of packets, and a new packet enters the network if and only if another packet from the same source leaves the network at the same time.

Closed queueing networks resemble networks with flow control operating under heavy loads (see, e.g., Reiser [116]); flow control limits the number of packets from

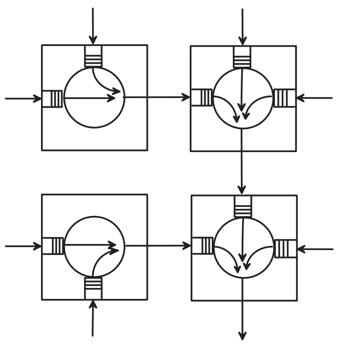


Figure 1.14: A mesh network where all traffic has the same destination.

the same source to a *maximum*, and heavy loads imply that served packets are quickly replaced by new packets from the same source, which is modelled by keeping the number of packets from the same source fixed. As networks on chips implement flow control as well, a closed network of polling stations can be used as a model for a network on chip where all traffic has the same destination, operating under a heavy load.

Polling systems

Polling systems have been the subject of numerous studies (for surveys, see [133,135, 142]) and have many applications, for example in telecommunications, transportation, and healthcare. Although single-station polling systems have been studied extensively, few attempts have been made to analyse *networks* of polling stations; one of the rare examples is a heavy-traffic study [115]. Below, we therefore give a brief introduction to single-station polling systems.

In polling literature it is common to speak of a server 'visiting' the various queues and serving the packets there. There are many different service disciplines that determine how many packets are served during a visit, such as exhaustive service, gated service, m_i-limited service, and Bernoulli service. With *exhaustive service*, the server serves a queue until it becomes empty. With *gated service*, each time the server visits a queue an imaginary gate is placed behind the last packet in the queue and the server only serves packets in front of that gate. With *m_i-limited service*, the server serves queue i until m_i packets have been served or queue i becomes empty, whichever happens first. With *Bernoulli service*, the server serves queue i again with probability q_i after a service completion there, and otherwise moves to another queue.

If the server moves to another queue, it might do so, for example, according to Markovian routing or cyclic routing. With *Markovian routing*, the server moves to queue j after service of queue i with probability r_{ij}. With *cyclic routing*, the server visits the queues in the natural order, i.e., after service of queue i the server moves to queue $i + 1 \mod N$. Cyclic routing is a special case of Markovian routing.

There is a remarkable distinction between service disciplines that so far have defied exact analysis of even the mean waiting time per queue (except for special cases like symmetric and 2-queue stations), such as 1-limited, and service disciplines for which various methods exist to obtain exact results, such as exhaustive and gated service. Resing [117] showed that service disciplines satisfying a so-called 'branching property' can be exactly analysed. This branching property states the following:

Property 1.4.2. *If the server arrives to queue i and finds k_i packets there, then during the course of the server's visit, all of these k_i packets are effectively replaced in an i.i.d. manner by an N-dimensional random population.*

For instance, with the gated service discipline, the packets left at the end of the visit of the server to queue i are the packets in queue $j \neq i$ that were present at the beginning of the visit to queue i plus the packets that arrived during the service of queue i. All packets present in queue i at the beginning of the visit to queue i

will have been removed when the server ends its visit. This can also be viewed as a replacement of every packet in queue i by the packets arriving during its service, i.e., by an i.i.d. N-dimensional random population.

In contrast, with the 1-limited service discipline *one* packet in queue i is replaced by packets arriving during its service. All other type i packets are left unchanged (i.e., replaced by one type i packet), so the 1-limited service discipline does not satisfy the branching property.

For service disciplines satisfying the branching property, it is shown in [117] that the number of packets in different queues, embedded at time points where the server visits queue 1, constitutes a multi-type branching process (MTBP) with immigration. Furthermore, it is mentioned that the class of MTBPs is one of the exceptional classes of multi-dimensional Markov chains for which the equilibrium distribution can be determined.

This at least partially explains why methods exist to obtain mean queue lengths (and thus mean waiting times) for exhaustive and gated service disciplines (for an overview of such methods, see, e.g., [147]). Nevertheless, even for these service disciplines, the mean waiting time per queue is, apart from special cases such as symmetric systems, not given explicitly but in terms of a matrix inverse, infinite product, or a solution to a set of equations.

The round robin scheduler of switches in networks on chips corresponds to the cyclic 1-limited service discipline, which implies that no exact results are known for siq-switches. There are, however, many approximations for 1-limited polling systems, such as that of Boxma and Meister [34], Levy and Groenendijk [68], and many others. This will be discussed in more detail in Chapter 6.

1.5 Key results and organisation of the thesis

In this section, we describe the key results of the thesis and we give an overview of how they are organised. Throughout this thesis, our focus is on the analysis of throughput and mean end-to-end delays. The throughput is a measure for how much data can be transmitted across the network and the delay is a measure for how long it takes to transmit data. Both are important measures for the performance of networks on chips and they need to be well understood.

In the first part of the thesis (Chapters 2 and 3), we focus on the analysis of the single-switch model. In Chapter 2, we first study a *uniform* packet switch with packets of size 1. Such switches have been analysed in literature under the assumption that N tends to infinity. However, in networks on chips, and in particular those with the mesh topology, switches often have only a few queues and we show that the known asymptotic analyses lead to inaccurate results for small switches. We approximate the mean waiting time in a switch by that in a $Geo/Geo/1$ queue and we show that this approximation is more accurate for small switches than the known asymptotic ones.

In Chapter 2 we furthermore study a small network of a *uniform* switch and network interfaces modelled by single server queues. Packets of fixed size K arrive to these single server queues and are then transmitted to the switch flit-by-flit. We extend the $Geo/Geo/1$ approximation to this case. The key argument in this extension is that the beginnings of packet transmissions become approximately periodic as the load increases, which reduces K to a time-scaling factor. This observation is the main motivation to consider single-flit packets from then on, and to disregard network interfaces. The analysis of Chapter 2 illustrates that this assumption has a high reward in terms of simplicity at only a small cost in terms of accuracy.

In Chapter 3, we consider a *non-uniform* switch with unit packet sizes and we extend our $Geo/Geo/1$ approximation to this case. The main difficulty in extending the approximation is that for given arrival rates, some queues might be stable while others are not. By further developing a heuristic approach proposed by Ibe and Cheng [72], we obtain a very accurate approximation of the throughput and saturation loads (the loads for which queues become unstable). Using the saturation load and throughput approximation, we can indeed extend our $Geo/Geo/1$ approximation to the non-uniform case. We also apply the approximation to two models with correlated traffic: The first one has correlation between arrivals (i.e., if an arrival occurs this time slot, it is more likely that an arrival will occur in the next time slot as well), and the second one has correlation between destinations (i.e., two consecutive packets are more likely to have the same destination).

In the second part of the thesis (Chapters 4, 5, 6, and 7), we consider networks of polling stations. Although these models are primarily motivated by networks on chips, the range of applications for which they are suitable extends far beyond networks on chips. This is reflected by the fact that we use the term 'node' rather than switch, and by the fact that we consider other service disciplines than 1-limited as well.

One of the main results of this part of the thesis is that we show that concentrating tree *networks* of polling stations can be reduced to *single-station* polling systems, while preserving information on queue lengths and waiting times. Most importantly, this *reduction theorem* makes it possible to analyse networks of polling systems through the use of single-station results.

The condition under which this reduction theorem holds is that the last node of the network (the sink) must use a so-called *HoL-based* service discipline. For the precise definition of HoL-based we refer to Chapter 4, but the definition entails that the server decides which packet it is going to serve at time t only based on whether queues are empty or non-empty at times $t, t-1, \ldots, t-M$ for an arbitrary finite M. It may not, for instance, take queue lengths into account. Service disciplines such as longest/shortest queue first are thus not HoL-based.

The class of HoL-based service disciplines includes - but is not limited to - the Bernoulli and m_i-limited service disciplines. Furthermore, if the server decides to select one of the other non-empty queues, it may do so according to some fixed order (e.g., a cyclic order) or according to Markovian routing. Exhaustive service is a special case of Bernoulli service, and a limiting case of m_i-limited, namely $m_i \to \infty$. The cyclic 1-limited service discipline used in switches is a special case of both.

The reduction theorem is proved in Chapter 4. In Chapter 5, we apply the reduction theorem of Chapter 4 to *all* nodes in a network. By making an additional approximation assumption, we obtain an approximation of the mean end-to-end delay per source. This approximation is derived for general HoL-based service disciplines, and its accuracy is studied for the cyclic 1-limited service discipline. We furthermore apply the approximation to a network on chip consisting of four switches in a mesh topology and we show that the reduction theorem can be used to obtain the mean end-to-end delay per source exactly in trees with a certain symmetry property.

The approximation of Chapter 5 requires the calculation of mean waiting times in single-station polling systems. In Chapter 5, we use a known approximation to compute these, namely the approximation of Boxma and Meister [34]. In Chapter 6 we derive a new approximation of the queue length distribution (from which mean waiting times follow) in single station polling systems. We do so for a large subclass of HoL-based service disciplines, namely that of Bernoulli service combined with Markovian routing, which contains the cyclic 1-limited service discipline as a special case. The approximation is found to be very accurate in general, and in particular for the cyclic 1-limited service discipline.

We study closed networks of polling stations in Chapter 7. Our study focuses on the effects of flow control on fairness in the network, i.e., on the division of throughput over packets from different sources. We model the network as a Markov chain and derive the exact throughput division for polling systems with the random polling service discipline (with random polling, the server serves every queue with a fixed probability, independently of what happened in previous time slots). In addition to this, we obtain the exact throughput division for polling systems with two queues and Bernoulli service and Markovian routing, of which random polling and 1-limited are a special case. The results from our analysis reveal that the division of throughput is steered by an interaction between service disciplines, buffer sizes, and the flow control mechanism. An additional numerical study sheds more light on the specifics of this interaction.

Chapters 2 until 6 are based on published papers: Chapter 2 is based on [13], Chapter 3 on [18], Chapter 4 on [15], Chapter 5 on [14], and Chapter 6 on [17]. The material of Chapter 7 has not yet been published, but a paper has been submitted [16].

CHAPTER 2

Uniform switches

In this chapter we analyse a network consisting of an $N \times N$ switch and N network interfaces modelled by single-server queues. Packets of fixed size K arrive to the network interfaces according to Bernoulli arrival processes and are transmitted to the switch flit-by-flit. We assume traffic in the network is uniform, i.e., all arrival rates are the same and all destinations are equally likely.

An important special case arises if $K = 1$; in this case, the network reduces to only a uniform $N \times N$ switch with Bernoulli arrivals. As discussed in Section 1.4.1, such switches have been analysed under the assumption that N tends to infinity. We show that these asymptotic models yield inaccurate results if they are applied to small switches. Furthermore, we propose a new approximation of the mean waiting time of packets that is specifically geared towards small switches. The key assumption in this approximation is that the time spent by a packet in the first position of a queue is geometrically distributed. As a result, the mean sojourn time of a packet in the switch is approximated by that in a $Geo/Geo/1$ queue. This approximation is compared to known asymptotic approximations, and shown to be more accurate for small N.

For the case where $K > 1$, we have batch arrivals at the single server queues representing the network interfaces. These batches arrive to the switch flit-by-flit, which is an arrival process of a kind that is sometimes known as 'train arrivals'. We analyse this network with train arrivals and extend our mean sojourn time approximation to the situation with $K > 1$. Using this approximation, we can also approximate the mean total sojourn time in the network.

2.1 Model

We consider a network on chip consisting of N network interfaces (NIs) and an $N \times N$ switch, where each NI is connected to a unique input port of the switch, as depicted in Figure 2.1. We assume that packets of fixed size K arrive at the NIs according to i.i.d. Bernoulli processes with parameter λ. Recall that the size of packets is measured in flits, where a flit is precisely the amount of data that can be transmitted in one time slot. The switch uses wormhole routing, which entails that packets are transmitted completely before the switch transmits another packet through the same output port.

Packets arriving to the network are stored in the NIs and the flits are sent to the switch one-by-one. An NI can thus be seen as a queue with unit service times and batch arrivals; a $Geo^X/D/1$ queue. In a system with batch arrivals, multiple entities requiring service (i.e., flits) arrive simultaneously. The superscript X denotes the size of the batch that arrives *every* time slot (with $\mathbb{P}(X = 0) > 0$). The interarrival times of non-zero batches are geometrically distributed with parameter $\mathbb{P}(X > 0)$.

Because the switch transmits flits, it can be seen as a server as well. The service time of a flit in the switch is defined as the time the flit spends in the HoL-position. The input process of the switch is the output process of the $Geo^X/D/1$ queue, which is an on-off process, whose on-period is equal to the busy period of the $Geo^X/D/1$ queue. Arrival processes of this form are sometimes called 'train arrivals' [149].

The path of an arbitrary packet across the network can thus be modelled as a discrete-time tandem network with two nodes. The first node represents the NI and is a $Geo^X/D/1$ queue with unit service times. The second node represents the switch and is a discrete-time $./G/1$ with an unknown service time distribution.

If the packet size $K = 1$, the arrival process at the switch simplifies to a Bernoulli process. Uniform packet switches with Bernoulli arrivals have been studied extensively under the assumption that N tends to infinity. We introduce a new $Geo/Geo/1$ approximation and show that it is more accurate than the asymptotically exact analysis of Karol et al. [77] and the asymptotic approximation of Kim et al. [80]. The latter approximation is also based on the assumption that $N \to \infty$, but Kim et al. make an additional approximation assumption that the number of consecutive output conflicts follows a geometric distribution.

For the case $K > 1$, we observe a certain periodicity in service times. This

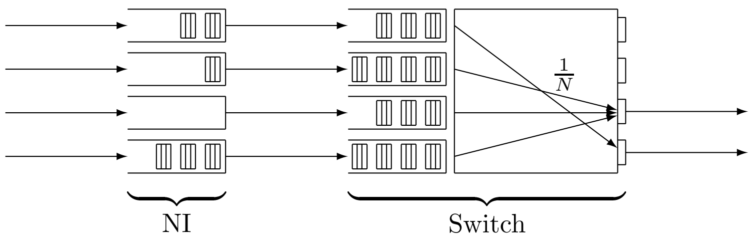


Figure 2.1: The network model of this chapter with $N = 4$ NIs.

periodicity is explained by the fact that service beginnings at multiple queues tend to occur simultaneously as the arrival rate increases. We extend our approximation to the case $K > 1$ by regarding K as a time-scaling factor. We furthermore analyse the network and approximate the mean total sojourn time in this network using the extension of our *Geo*/*Geo*/1 approximation.

The mean sojourn time $\mathbb{E}[S]$ of the $Geo^X/G/1$ is an important quantity in this chapter. It can be found using $\mathbb{E}[S] = \mathbb{E}[W] + \mathbb{E}[B]$, where $\mathbb{E}[B]$ is the mean service time and $\mathbb{E}[W]$ the mean waiting time, which is given by Equation (1.52a) of Takagi [134]:

$$\mathbb{E}[S] = \frac{\rho}{1-\rho}\left(\frac{\mathbb{E}[B^2]}{2\,\mathbb{E}[B]} - \frac{1}{2}\right) + \frac{1}{1-\rho}\left(\frac{\mathbb{E}[X^2]}{2\,\mathbb{E}[X]} - \frac{1}{2}\right)\mathbb{E}[B] + \mathbb{E}[B], \tag{2.1.1}$$

where $\rho = \mathbb{E}[X]\,\mathbb{E}[B]$ is the load.

The organisation of the remainder of this chapter is as follows: Section 2.2 is devoted to approximations for the case $K = 1$. In Section 2.2.1 and 2.2.2 we describe the approximations of Karol et al. and Kim et al. respectively. In Section 2.2.3 we introduce our new *Geo*/*Geo*/1 approximation. In Section 2.3 we extend this approximation to the case $K > 1$. The tandem network is analysed in Section 2.4.

The second part of this chapter contains more empirical results. First, we analyse the performance of our approximation in Section 2.5 and we compare our approximation to the asymptotic models. The assumption that the service time is geometrically distributed is investigated in more depth in Section 2.6. Finally, we present the conclusions of our research in Section 2.7.

2.2 Approximations for $K = 1$

If $K = 1$, the network interfaces delay all packets for precisely one time slot, so the arrival process to the switch is a Bernoulli process. The network is thus reduced to only the uniform switch with Bernoulli arrivals, as displayed in Figure 2.2.

In Sections 2.2.1 and 2.2.2, we briefly review the asymptotic approximations of Karol, Hluchyj, and Morgan [77] and Kim, Kim, and Lee [80], referred to as the KHM and KKL model respectively. The main difference between these two models is that in the KHM model the first two moments of the service time are determined exactly, whereas in the KKL model, a geometric service time distribution is assumed

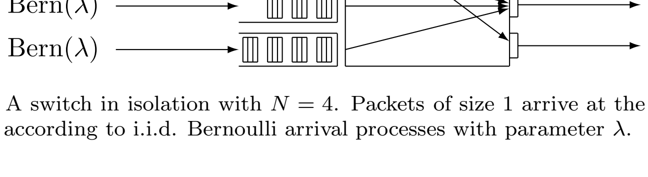


Figure 2.2: A switch in isolation with $N = 4$. Packets of size 1 arrive at the input queues of the switch according to i.i.d. Bernoulli arrival processes with parameter λ.

and fitted to the mean of the KHM model. In Section 2.2.3 we devise our $Geo/Geo/1$ approximation.

2.2.1 The KHM approximation

The crucial observation of Karol et al. [77] is that if $N \to \infty$ the number of packets with the same destination arriving each time slot at the *HoL-positions* of the input queues follows a Poisson(λ) distribution. Because packets are selected for transmission at random, the time spent by a packet in the HoL-position of an input queue (i.e., the service time) is in distribution equal to the sojourn time in a discrete-time queue with Poisson arrivals, service in random order, and unit service times. If we let $\widetilde{X} \sim$ Poisson(λ) be the batch size and $\widetilde{S}$ the sojourn time in the latter $Geo^{\widetilde{X}}/D/1$ queue, we thus have

$$B \overset{d}{=} \widetilde{S}, \tag{2.2.1}$$

where $\overset{d}{=}$ denotes equality in distribution.

To calculate the mean sojourn time in the *switch*, we need to determine the first and second moment of the service time B. Karol et al. provide a numerical procedure to find the distribution of $\widetilde{S}$, which in particular allows us to determine $\mathbb{E}[B^2] = \mathbb{E}[\widetilde{S}^2]$. Furthermore, the mean sojourn times with service in random order and FIFO are equal, so it follows from Equation (2.1.1) that

$$\mathbb{E}[B] = \mathbb{E}[\widetilde{S}] = \frac{\widetilde{\rho}}{1-\widetilde{\rho}}\left(\frac{\mathbb{E}[\widetilde{B}^2]}{2\,\mathbb{E}[\widetilde{B}]} - \frac{1}{2}\right) + \frac{1}{1-\widetilde{\rho}}\left(\frac{\mathbb{E}[\widetilde{X}^2]}{2\,\mathbb{E}[\widetilde{X}]} - \frac{1}{2}\right)\mathbb{E}[\widetilde{B}] + \mathbb{E}[\widetilde{B}], \tag{2.2.2}$$

where $\widetilde{\rho} = \mathbb{E}[\widetilde{X}]\,\mathbb{E}[\widetilde{B}]$ and $\widetilde{B} \sim$ Det(1). Furthermore, $\widetilde{X} \sim$ Poisson(λ), which implies $\mathbb{E}[\widetilde{X}] = \lambda$ and $\mathbb{E}[\widetilde{X}^2] = \lambda(1+\lambda)$. Hence, (2.2.2) simplifies to

$$\mathbb{E}[B] = \frac{\lambda}{2(1-\lambda)} + 1. \tag{2.2.3}$$

Finally, we observe that, due to the Bernoulli arrival processes, the switch is a $Geo/G/1$ queue, which is a special case of the $Geo^X/G/1$ queue with

$$X = \begin{cases} 0, & \text{w.p. } 1-\lambda, \text{ and} \\ 1, & \text{w.p. } \lambda, \end{cases}$$

and therefore $\mathbb{E}[X^2] = \lambda$. We infer from Equation (2.1.1) that the mean sojourn time in the switch is thus given by:

$$\mathbb{E}[S] = \frac{\rho}{1-\rho}\left(\frac{\mathbb{E}[B^2]}{2\,\mathbb{E}[B]} - \frac{1}{2}\right) + \mathbb{E}[B], \tag{2.2.4}$$

where $\rho = \lambda\,\mathbb{E}[B]$, $\mathbb{E}[B]$ is as in (2.2.3) and $\mathbb{E}[B^2]$ can be determined using the numerical procedure of Karol et al. [77].

2.2.2 The KKL approximation

Kim et al. [80] also model the switch as a $Geo/G/1$ queue for $N \to \infty$, but they assume a geometric service time distribution with the same mean service time as found by Karol et al. [77]. That is, they assume that any queue of the switch is a $Geo/Geo/1$ queue with service rate $\mu = 1/\mathbb{E}[B]$, with $\mathbb{E}[B]$ as in Equation (2.2.3).

Due to the geometric service time distribution, we have:

$$\mathbb{E}[B^2] = \frac{2-\mu}{\mu^2}.$$

The mean sojourn time in the switch now follows from Equation (2.1.1):

$$\mathbb{E}[S] = \frac{\rho}{1-\rho}\left(\frac{1}{\mu} - 1\right) + \frac{1}{\mu} = \frac{(1-\lambda)(2-\lambda)}{\lambda^2 - 4\lambda + 2}.$$

Note that the model is based on a non-saturated switch, which means that the results are only valid for $\lambda < 2 - \sqrt{2}$. This is reflected in the denominator of $\mathbb{E}[S]$, because $\lambda = 2 - \sqrt{2}$ is one of its zeros.

2.2.3 Geometric approximation

In this subsection, we introduce our new $Geo/Geo/1$ approximation. The key step in this approximation is that, like in the approximation of Kim, Kim, and Lee [80], we assume that the service time is geometrically distributed. However, instead of fitting the parameter of this geometric distribution to the mean found by Karol, Hluchyj, Morgan [77], we use a quadratic approximation of the service rate based on a light traffic limit and the saturation throughput. Because our model is not based on the limit of $N \to \infty$ we find a different mean service time and obtain better results for small switches.

The throughput of a switch is defined as the expected number of served flits per time slot, divided by N. The saturation throughput λ_{sat} is defined as the throughput in saturation, which is an overload situation where all queues are always non-empty. If the loads are high enough, i.e., if $\lambda \geq \lambda_{sat}$, the service rate is equal to the saturation throughput.

For small N, the saturation throughput can be determined using a Markov chain approach, as was briefly discussed in Section 1.4.1. For more details on how to compute the saturation throughput in uniform switches, the reader is referred to [12, 30], and to Chapter 3, where we describe a more general Markov chain primarily aimed at *non-uniform* switches. Table 2.1 shows the saturation throughput for some values of N. For large N, the Markov chain approach becomes intractable and we have to resort to simulation approximations or to its limiting value $2 - \sqrt{2} \approx 0.586$.

It remains to determine the service rate in light traffic; we consider $\lambda > 0$ small and we neglect $\mathcal{O}(\lambda^2)$ terms. By doing so, we can determine the entire light traffic service time distribution, which also gives us the light traffic service rate.

Consider an arbitrary time slot t and an arbitrary packet which we tag. Suppose that the tagged packet arrives at a non-empty switch. Because there is at least one

N	λ_{sat}	N	λ_{sat}
1	1	7	0.6238
2	0.75	8	0.6184
3	0.6825	9	0.6146
4	0.6552	10	0.6116
5	0.6399	11	0.6091
6	0.6302	∞	0.586

Table 2.1: Saturation throughputs for several values of N.

packet present from slot $t-1$, there must have been at least two packets present in that time slot. This implies that at some point in time there must have been two simultaneous arrivals. Since this happens with a probability of $\mathcal{O}(\lambda^2)$, we may ignore the situation in which a packet arrives at a non-empty switch.

So consider a tagged packet arriving at an empty system. The tagged packet is almost always switched except if another packet arrives simultaneously with the same destination and wins contention. Note that the probability of two or more *other* arrivals is $\mathcal{O}(\lambda^2)$, which we neglect. The probability that one other packet arrives is $(N-1)\lambda+\mathcal{O}(\lambda^2)$ because there are $N-1$ remaining input queues and at each queue an arrival happens with probability λ. The two packets have the same destination with probability $\frac{1}{N}$, in which case the tagged packet is *not* switched with probability $\frac{1}{2}$. Multiplying these probabilities gives us that the tagged packet is switched with probability $1-\frac{1}{2}\frac{N-1}{N}\lambda+\mathcal{O}(\lambda^2)$. Since another arrival in the next time slot would induce another factor λ, the probability that the tagged packet is not switched in the next time slot is of $\mathcal{O}(\lambda^2)$. Altogether we thus obtain

$$B=\begin{cases}1 & \text{w.p. } 1-\frac{1}{2}\frac{N-1}{N}\lambda+\mathcal{O}(\lambda^2),\\ 2 & \text{w.p. } \frac{1}{2}\frac{N-1}{N}\lambda+\mathcal{O}(\lambda^2),\\ k\geq 3 & \text{w.p. } \mathcal{O}(\lambda^2),\end{cases}\tag{2.2.5}$$

which implies that

$$\frac{1}{\mathbb{E}[B]}=1-\frac{1}{2}\frac{N-1}{N}\lambda+\mathcal{O}(\lambda^2).\tag{2.2.6}$$

The service rate approximation we propose is exact in light traffic and if $\lambda\geq\lambda_{sat}$, and continuous at λ_{sat}. These requirements result in the following quadratic approximation:

$$\mu(\lambda)=\begin{cases}1-\frac{1}{2}\frac{N-1}{N}\lambda+\left(\left(1+\frac{1}{2}\frac{N-1}{N}\right)\frac{1}{\lambda_{sat}}-\frac{1}{\lambda_{sat}^2}\right)\lambda^2, & \text{for } 0\leq\lambda<\lambda_{sat},\\ \lambda_{sat}, & \text{for } \lambda\geq\lambda_{sat}.\end{cases}\tag{2.2.7}$$

The sojourn time in the switch is approximated by that in a *Geo*/*Geo*/1 queue with arrival rate λ and service rate $\mu(\lambda)$ (and hence $\rho=\lambda/\mu(\lambda)$):

$$\mathbb{E}[S]=\frac{\rho}{1-\rho}\left(\frac{1}{\mu(\lambda)}-1\right)+\frac{1}{\mu(\lambda)}=\frac{1-\lambda}{\mu(\lambda)-\lambda}.\tag{2.2.8}$$

Remark 2.2.1. Perhaps the accuracy of $\mu(\lambda)$ can be improved by taking higher order terms into account. Most importantly, however, this would prevent us from considering only flits that arrive at an empty system, which complicates the analysis. Although incorporating higher order terms constitutes an interesting research option, the present approximation is sufficiently accurate for our purposes, as is shown in Section 2.5.

2.3 Service time approximation for $K > 1$

In this section we extend our approximation to the case $K > 1$. From the analysis of simulation results, we infer that there is a certain periodicity in the service time distribution. This periodicity is perhaps best explained in Figure 2.3 where the service time distribution is plotted for $N = 4$, $K = 6$ and $\lambda = 0.01, 0.06, 0.10$. Especially for $\lambda = 0.1$, there are large peaks for $i = 1, 7, 13, 19, \ldots$, while for $i = 2, \ldots, 6$, $i = 8, \ldots, 12$, $i = 14, \ldots, 18$, and $i = 20, \ldots, 24$, the service probabilities seem to be uniform. Note that on average λK flits arrive at the switch per time slot, so $\lambda = 0.1$ implies that the arrival rate is already close to the saturation value of 0.655 (see Table 2.1).

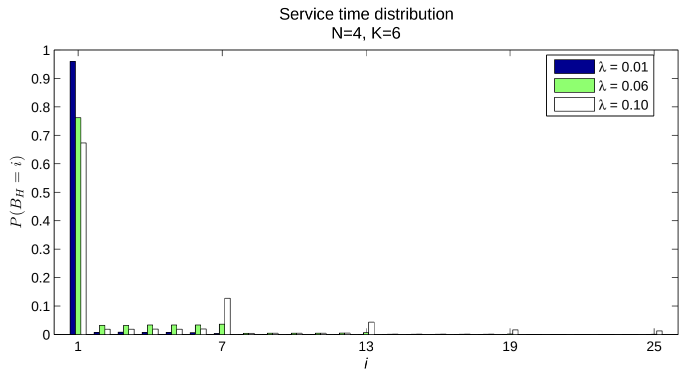
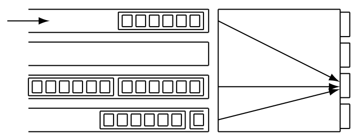


Figure 2.3: The distribution of the header service time B_H.

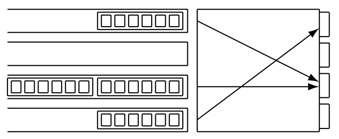
Visual simulation output provided a very good explanation for this periodicity. Because the packet sizes are the same for all inputs, the service beginnings at multiple input queues occur at the same time, in a periodic manner with period K. We call this phenomenon 'alignment' of packets. Once alignment occurs, it can only be broken if one of the queues gets empty, which explains why the phenomenon is more apparent for large λ. The alignment phenomenon is also described by Figure 2.4.

The alignment principle also implies that the throughput does not depend on K. If the load exceeds the saturation load the packets always remain aligned because the queues never become empty, so K is only a time-scaling factor.

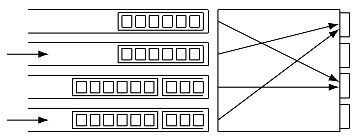
We extend the geometric approximation by ignoring the possibility that pack-



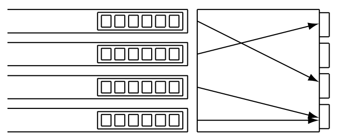
(a) Time slot t. One packet is being switched, one was already waiting and one new packet is arriving.



(b) Slot $t+1$. The packet at port 4 has been served, a new packet attains the HOL-position.



(c) Slot $t{+}4$. Two new arrivals. All packets are now 'aligned'.



(d) Slot $t+7$. The packets remain aligned until one of the input queues gets empty.

Figure 2.4: Alignment of packets. In the pictures the packets arrive at the switch in their entirety but this does not fundamentally change the alignment concept.

ets are not aligned. We assume that the number of successive output conflicts is geometrically distributed with parameter $\mu_K(\lambda)$, i.e., a header is transmitted with probability $\mu_K(\lambda)$ and it is not transmitted with probability $1-\mu_K(\lambda)$. If a header is not transmitted, it has to wait for an additional K time units, until the other packet has fully completed its service. After this, the header is again switched with probability $\mu_K(\lambda)$ and so on.

Recall that wormhole routing is used, so non-header flits are always immediately switched. The distribution of B_H, the service time of a header, is approximated as follows:

$$\mathbb{P}(B_H = mK+1) = \mu_K(\lambda)(1-\mu_K(\lambda))^m \qquad \text{for } m = 0, 1, 2, \dots. \tag{2.3.1}$$

Furthermore, we set $\mu_K(\lambda) = \mu(\lambda K)$, with μ as in Eq. (2.2.7). This choice is also motivated by the observation that if λ is large, K is only a time-scaling factor.

2.4 Network analysis

In this section we analyse the network under the assumption that $K > 1$. In this case, the input process of the switch is no longer Bernoulli and the switch can therefore no longer be seen as a $Geo^X/G/1$ queue. Instead, there are train arrivals where each train has a length equal to the busy period in a $Geo^X/D/1$ queue. The number of empty slots between two successive trains follows a geometric distribution with parameter λ.

This particular output process clearly complicates the analysis of the network. This complication, however, can be circumvented by first regarding an artificial model in which packets arrive in their entirety at the switch. This model is studied in Section 2.4.1. In Section 2.4.2, the results of Section 2.4.1 are combined with the service time approximation of Section 2.3 in order to approximate the total mean sojourn time in the network.

2.4.1 Arrivals at the switch

In this section we consider the artificial situation in which packets arrive at the switch in their entirety. In other words, each time slot a packet of size $K > 1$ arrives at each input queue according to a Bernoulli process with parameter λ, so a single queue of the switch can be seen as a $Geo^X/G/1$ queue. All performance measures of this artificial network are denoted by tildes above the normal letters.

In Section 2.3 we derived an approximation for the service time distribution of a header. We cannot, however, directly substitute this distribution in the formula for the mean sojourn time of a $Geo/G/1$ queue, because service times of headers and non-headers differ. Service times of non-headers are always 1, whereas the header has to win contention; the header is thus a special first customer in a batch.

This difficulty can be resolved by viewing every packet as a single entity requiring service. The service time $\widetilde{B}$ of a packet is equal to the time it takes the header to win the output conflict and an additional $K-1$ time slots for the non-header flits to be transmitted, i.e. $\widetilde{B} = B_H + (K-1)$. From the approximation of the distribution of B_H (Eq. (2.3.1)), we obtain:

$$\mathbb{P}(\widetilde{B} = mK) = \mathbb{P}(B_H = (m-1)K+1) = \mu_K(\lambda)(1-\mu_K(\lambda))^{m-1} \qquad \text{for } m = 1, 2, \ldots, \tag{2.4.1}$$

which implies $\mathbb{E}\,\widetilde{B} = K/\mu_K(\lambda)$, and

$$\mathbb{E}[\widetilde{B}^2] = \frac{2-\mu_K(\lambda)}{(\mu_K(\lambda))^2}K^2.$$

The mean sojourn time $\mathbb{E}\,\widetilde{S}$ of a super-customer can now be found by applying Equation (2.1.1):

$$\mathbb{E}[\widetilde{S}] = \frac{\lambda K}{\mu_K(\lambda) - \lambda K}\left(\frac{K}{\mu_K(\lambda)} - \frac{1}{2}(K+1)\right) + \frac{K}{\mu_K(\lambda)}. \tag{2.4.2}$$

2.4.2 Arrivals at the NI

We now study the network with arrivals at the NI. We obtain the total mean sojourn time of packets in the network, and the mean sojourn time of packets in the switch. The sojourn time of a packet in the network, denoted by T, is defined as the time between the arrival of a packet at the NI and the departure of its last flit from the switch. Likewise, the sojourn time of a packet in the switch, S, is defined as the time between the arrival of the first flit and the departure of the last.

With sample-path wise identical arrivals, services of packets at the switch always start and end precisely one time slot later in the network with arrivals at the NI than in the artificial network with arirvals at the switch (see Friedman [57] for more details). In particular, this implies that the sojourn time in the network is equal to the sojourn time in the switch in the artificial situation, plus one, i.e., $T = \widetilde{S} + 1$ with $\widetilde{S}$ as in Section 2.4.1 . It thus follows that

$$\mathbb{E}[T] = \mathbb{E}[\widetilde{S}] + 1 = \frac{\lambda K}{\mu_K(\lambda) - \lambda K}\left(\frac{K}{\mu_K(\lambda)} - \frac{1}{2}(K+1)\right) + \frac{K}{\mu_K(\lambda)} + 1. \tag{2.4.3}$$

To obtain the mean sojourn time of packets in the switch, we observe that its sojourn time in the network consists of the sojourn time of the header in the NI, and the sojourn time in the switch. We thus obtain

$$T = S_{H,NI} + S, \tag{2.4.4}$$

where $S_{H,NI}$ is the sojourn time of a header in the NI.

The mean sojourn time of a header in the NI is equal to the mean sojourn time of the entire packet in the NI (i.e., the time between the packet arrival and departure of the last flit), minus $K-1$. In order to determine the mean sojourn time of packets in the NI, we view the NI itself as a $Geo/D/1$ queue with service times equal to K. Note that this queue is a special case of the $Geo^X/G/1$ queue with

$$X = \begin{cases} 0, & \text{w.p. } 1-\lambda, \\ 1, & \text{w.p. } \lambda, \end{cases}$$

and $\mathbb{P}(B = K) = 1$. By subtracting $K - 1$ from Equation (2.1.1), we thus obtain:

$$\begin{aligned}\mathbb{E}[S_{H,NI}] &= \frac{\rho}{1-\rho}\left(\frac{\mathbb{E}[B^2]}{2\,\mathbb{E}[B]} - \frac{1}{2}\right) + \frac{1}{1-\rho}\left(\frac{\mathbb{E}[X^2]}{2\,\mathbb{E}[X]} - \frac{1}{2}\right)\mathbb{E}[B] + \mathbb{E}[B] - (K-1) \\ &= \frac{\lambda K(K-1)}{2(1-\lambda K)} + 1.\end{aligned}$$

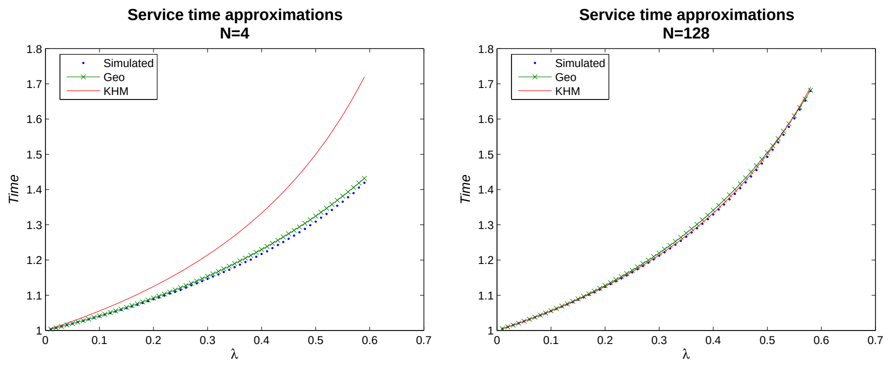
Finally, we apply Equation (2.4.4) to obtain

$$\begin{aligned}\mathbb{E}[S] &= \mathbb{E}[T] - \mathbb{E}[S_{H,NI}] \\ &= \frac{\lambda K}{\mu_K(\lambda) - \lambda K}\left(\frac{K}{\mu_K(\lambda)} - \frac{1}{2}(K+1)\right) + \frac{K}{\mu_K(\lambda)} - \frac{\lambda K(K-1)}{2(1-\lambda K)}.\end{aligned} \tag{2.4.5}$$

2.5 Approximation comparison

In this section, we analyse the accuracy of the approximations of the previous sections. First, we analyse the mean service time approximation, then the mean sojourn time approximation for $K = 1$, and finally the mean sojourn time approximation for $K > 1$.

In Figure 2.5 we display the approximation of the mean service time of the KHM approximation and our $Geo/Geo/1$ approximation versus simulation results. Recall that the KHM and KKL approximations of the mean service time are the same. Clearly, our mean service time approximation performs much better than the other



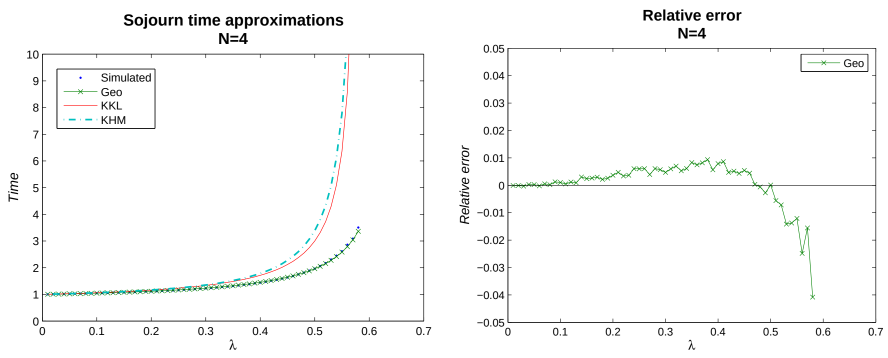
(a) $N = 4$ **(b)** $N = 128$

Figure 2.5: Mean service time approximations for $N = 4$ and $N = 128$, both with $K = 1$.

approximations if N is small. This result is not very surprising, as our approach is based on small N rather than the limit of $N \to \infty$. Furthermore, for $N = 128$ our approximation is equally accurate as the asymptotic approximation.

The approximations of the mean sojourn time are compared with simulation results in Figure 2.6a. From this figure it is again clear that our $Geo/Geo/1$ approximation outperforms the other approximations if $N = 4$. In Figure 2.6b, the relative error of our approximation is plotted. Until the system approaches saturation, there is a maximum relative error of roughly 1%.

We also see that the asymptotic approximations can have large errors for loads close to the saturation throughput. After all, the asymptotic saturation throughput



(a) Approximations **(b)** Relative error

Figure 2.6: Mean sojourn time approximations and the relative error of our approximation for $N = 4$.

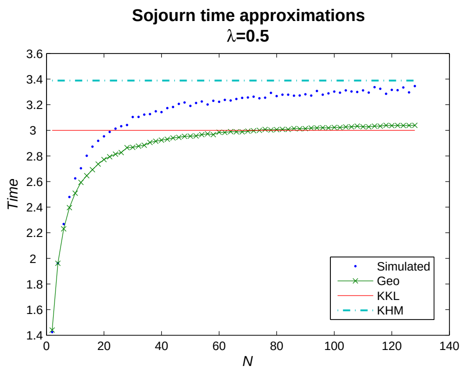


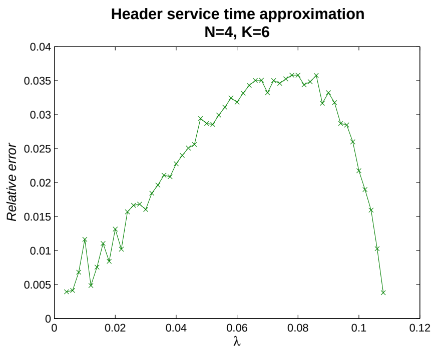
Figure 2.7: Mean sojourn time approximations with λ fixed. Note that both the KKL and the KHM model are based on $N \longrightarrow \infty$, which explains the horizontal line.

is $2 - \sqrt{2} \approx 0.586$, whereas for $N = 4$ the saturation throughput is 0.655 (see Table 2.1). This difference leads to very large errors if λ is close to saturation. For instance, for $\lambda = 0.5$ the mean sojourn time approximation of Karol et al. has a relative error of roughly 70%. By choosing λ even higher, the error can be made arbitrarily large.

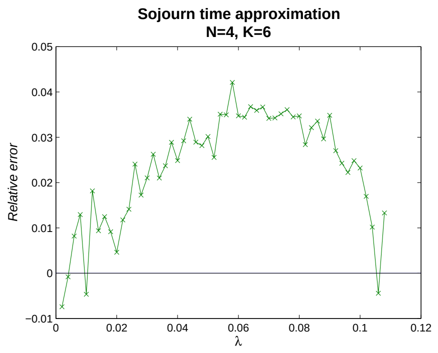
In order to get some insight in the role of N, we show the sojourn time approximations as a function of N with $\lambda = 0.5$ in Figure 2.7. Most importantly, we can conclude that our model is a considerable improvement over the other models for small N. For large N, say $N > 60$, the asymptotic analysis of Karol et al. is the most accurate.

Remarkably, our approximation of the mean *service* time is quite accurate for $N = 128$, whereas our mean *sojourn* time approximation gives a significant error. Apparently the error we make in assuming that the service time distribution is geometric becomes more important for larger N. This conclusion is also backed by the fact that the KKL approximation still deviates from simulation outcomes as N grows large, whereas the KHM approximation is asymptotically exact. In Section 2.6 we analyse the consequences of our assumption that the service time distribution is geometric in more detail.

Finally, we study the accuracy of our mean header service time and mean sojourn time approximation for the case $K > 1$. Our approximation still gives very good results: Figure 2.8a illustrates that the relative error of the mean header service time approximation compared to simulation outcomes is maximally 3.5%, and Figure 2.8b illustrates that the mean sojourn time approximation has a relative error of at most 4.5%.



(a) The relative error of the approximation of $\mathbb{E}\, B_H$.



(b) The relative error of the mean sojourn time approximation.

Figure 2.8: The relative error of the mean header service time and mean sojourn time approximations.

2.6 Validation of the geometric distribution

In this section we validate the assumption that the service time is geometrically distributed. It can rather easily be argued that the *actual* service time distribution is *not* geometric; the geometric distribution is memoryless, but the switching probability is not. For instance, if all HoL-positions are occupied with packets with the same destination, each packet has probability $\frac{1}{N}$ of being switched. If the newly arriving packet has a different destination, then the remaining packets are all switched in the next time slot with probability $\frac{1}{N-1}$, and so on. This implies that there is some dependency on the history of the process, yet the precise effect of this dependency is unclear.

In order to study to what extent the service time distribution deviates from a geometric distribution, we introduce the 'best geometric' approximation. In this approximation, we still approximate the sojourn time in the switch by that in a *Geo*/*Geo*/1 queue, but the service rate is chosen equal to $1/E[B]$, where $\mathbb{E}[B]$ is determined using simulation. This way, there are no additional errors from the quadratic interpolation, which provides insight in how well the service time distribution is approximated by a geometric distribution.

Figure 2.9 shows the service time distribution obtained by simulation, the service time distribution of our *Geo*/*Geo*/1 approximation (i.e., a $Geo(\mu(\lambda))$ distribution), and a geometric distribution with a rate equal to the service rate obtained by simulatio (i.e., the service time distribution of the 'best geometric' approximation). From this figure we may conclude that the service time distribution indeed seems roughly geometric.

To confirm this conjecture, we look at the first and second moments in Table 2.2. In addition to this, Table 2.2 shows the value of $\mathbb{E}[B^2]/2\,\mathbb{E}[B]$ which has a prominent influence on the mean sojourn time approximation. We conclude that the simulated values of these quantities lie closer to those of the *Geo*/*Geo*/1 approximation than

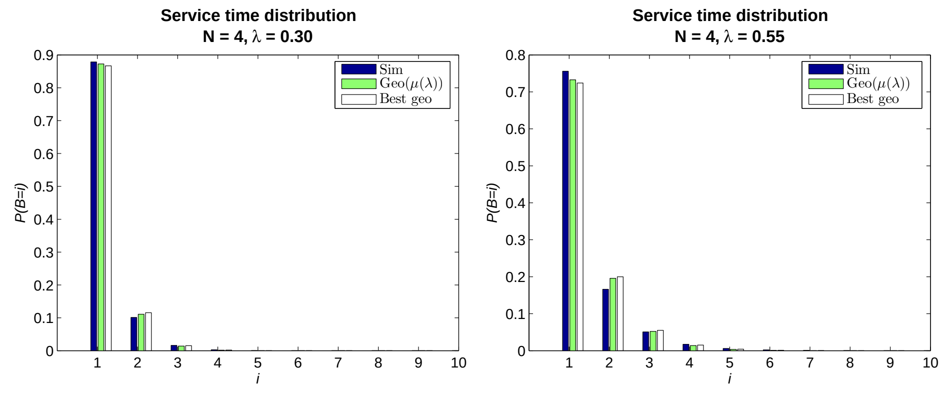


Figure 2.9: The service time distribution, a $Geo(\mu(\lambda))$ distribution, and the 'best geometric' approximation.

those of the 'best geometric' approximation. In other words, the approximation is more accurate if the quadratic interpolation is used than if the service rate is determined using simulation.

	Sim.	$Geo(\mu(\lambda))$	B.g.
$\mathbb{E}[B]$	1.365	1.381	1.365
$\mathbb{E}[B^2]$	2.471	2.435	2.361
$\frac{\mathbb{E}[B^2]}{2\,\mathbb{E}[B]}$	0.905	0.881	0.865

(a) $N = 4$

	Sim.	$Geo(\mu(\lambda))$	B.g.
$\mathbb{E}[B]$	1.493	1.505	1.493
$\mathbb{E}[B^2]$	3.336	3.024	2.964
$\frac{\mathbb{E}[B^2]}{2\,\mathbb{E}[B]}$	1.117	1.005	0.993

(b) $N = 128$

Table 2.2: The first and second moments of the service time, and $\frac{\mathbb{E}[B^2]}{2\,\mathbb{E}[B]}$ with $\lambda = 0.55$. 'B.g' stands for 'best geometric'.

Finally, we compare the mean sojourn time approximations in Figure 2.10. In this figure, we clearly see that for a large range of λ (say $\lambda \geq 0.35$), the $Geo/Geo/1$ approximation is more accurate than the 'best geometric' approximation. Apparently the error made in the service rate approximation compensates to some extent for the error in the geometric distribution assumption. The second moment of the service time distribution is larger than the second moment of a geometric distribution with the same mean. The mean service time is overestimated by $1/\mu(\lambda)$, resulting in a higher approximation of the second moment as well, which compensates for the underestimation caused by assuming a geometric service time distribution.

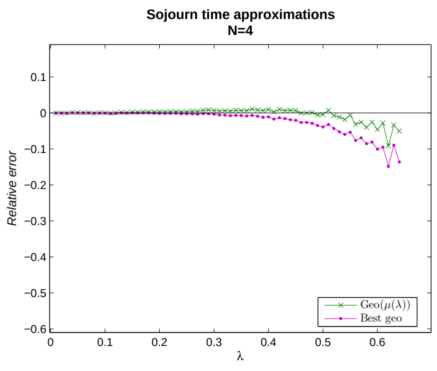


Figure 2.10: Relative errors of the 'best geometric' mean sojourn time approximation and the $Geo/Geo/1$ mean sojourn time approximation.

2.7 Conclusion

The mean sojourn time in a uniform $N \times N$ switch with Bernoulli arrival processes can be accurately approximated by that in a $Geo/Geo/1$ queue. The service rate of this queue can be approximated by $\mu(\lambda)$, a quadratic interpolation between the service rate in light traffic and saturation. For small switches, such as those in networks on chips, the $Geo/Geo/1$ approximation is a significant improvement over the approximations of Karol et al. [77] and Kim et al. [80] that are based on the assumption that $N \to \infty$.

The service rate approximation $\mu(\lambda)$ is very accurate in general, even for large N. Nevertheless, the error of the mean sojourn time approximation becomes larger if N increases.

If $K > 1$, we have train arrivals at the switch. This can be overcome by analysing an artificial model in which packets arrive in their entirety at the switch. In this artificial model we can determine the mean sojourn time and using that we can determine the mean sojourn time in our original network. Furthermore, if $K > 1$ we observe an alignment phenomenon, by which we mean that service beginnings of packets become periodic and simultaneous among queues. We can extend our $Geo/Geo/1$ approximation by regarding K as a time-scaling factor, and the resulting approximation has a small relative error.

The technique of regarding the packet size as a time-scaling factor can be applied similarly in the other models of this thesis, which is our main motivation for considering packets of unit size in the sequel. As the analysis of this chapter illustrated, such an assumption offers a great improvement in simplicity at only a small cost in accuracy.

Chapter 3

Non-uniform switches

In this chapter, we extend the *Geo*/*Geo*/1 approximation of Chapter 2 to *non-uniform* switches. The light traffic analysis of Chapter 2 can be extended in a quite straightforward manner. The key difficulty, however, is that for heavy loads, queues in non-uniform switches become unstable for different loads.

By adapting an idea of Ibe and Cheng [72], we obtain a very accurate approximation for the saturation loads (the loads for which queues become unstable). This approach also allows us to approximate the throughput of a queue given that it is unstable. Together with the light traffic analysis, we obtain an accurate approximation of the mean waiting time in switches. Our approximations are initially derived under the assumption of Bernoulli arrival processes, but they are also extended to models with correlated traffic. Their accuracy is extensively verified through comparison with simulation results.

3.1 Model

We consider a single input-queued switch with N input ports and M output ports operating in discrete-time. Each packet in queue i has destination j with probability p_{ij}, independently of all other packets. All buffers are infinitely large and each packet consists of one flit. The arrival rate to queue i is denoted by λ_i. A more schematic representation of the model can be found in Fig. 3.1.

We assume the switch uses the random order discipline, which means that if there are k packets with the same destination, one is selected for transmission, and all packets have probability $1/k$ of being selected, independently of what happened in previous time slots. We define the service time B_i of a packet in queue i as the time spent in the HOL-position of the queue. The service time is thus equal to the number of successive attempts to reach its destination, including the successful one.

This chapter is organised as follows: Section 3.2 is devoted to the analysis of a saturated switch as a Markov chain. This analysis is used in an approximation of the stability conditions in Section 3.3 and an approximation of the throughput of unstable queues in Section 3.4. The derivation of the mean waiting time approximation is given in Section 3.5. We perform an in-depth numerical analysis of one example with Bernoulli arrival processes in Section 3.6, and with more general (correlated) arrival processes in Section 3.7. We perform a large-scale numerical analysis of 100 examples with Bernoulli arrival processes in Section 3.8. Finally, we draw conclusions and discuss our results in Section 3.9.

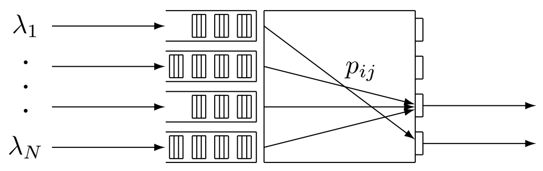


Figure 3.1: A schematic representation of the model of a switch. Packets arrive at rate λ_i to queue i of the switch, and have destination j with probability p_{ij}.

3.2 Saturated switch

A saturated switch is a switch of which all queues are always non-empty. As described in Section 1.4.1, saturated switches can be modelled by a Markov chain $D(t) = (D_1(t), \dots, D_N(t))$, where $D_i(t)$ is the destination of the HoL-packet of queue i in time slot $[t, t+1)$. The state space is given by $\Omega := \{1, \dots, M\}^N$ and the switch is said to be in state $x = (x_1, \dots, x_N)$ if the packet at queue i has destination x_i. In this section we descibe how the throughput of a saturated switch can be calculated using this Markov chain approach.

The transition probabilities $\mathcal{P}(x, y)$ of the Markov chain $D(t)$ can be determined straightforwardly by dividing the transitions into two steps: service completions and replacements of served packets by new packets. The state of the switch after the first step is described by 'intermediate' states, with zeros indicating empty input ports (i.e., a packet at that port has just been served). The second step consists of replacing zeros; a 0 at position i is replaced by a j with probability p_{ij}.

In a more formal setting, we define $A(x)$ to be the set of intermediate states reachable from x. Because each service possibility is equally likely, we get that:

$$\mathcal{P}(x,y) = \frac{1}{|A(x)|} \sum_{a \in A(x)} \prod_{i=1}^{N} \Big(1(a_i = 0) p_{i,y_i} + 1(a_i = y_i) \Big),$$

where $1(\cdot)$ is the indicator function. From $\mathcal{P}(x,y)$ we can determine the steady-state distribution $\pi(\cdot)$ of the Markov chain by using the fact that $\pi(\cdot)$ is the unique normalised solution of the system of equations

$$\pi(y) = \sum_{x \in \Omega} \pi(x) \mathcal{P}(x,y), \quad y \in \Omega.$$

For our purposes, the most important performance measure of the Markov chain is the steady-state throughput. The instantaneous throughput of queue i in state x is given by

$$\gamma_i(x) = \left(\sum_{k=1}^{N} 1(x_k = x_i) \right)^{-1}.$$

Here, $\sum_{k=1}^{N} 1(x_k = x_i)$ is the number of packets that have the same destination as the packet in queue i. The probability that the packet in queue i is selected for transmission is given by $\gamma_i(x)$ due to the random order of service. The steady-state throughput of queue i is given by

$$\gamma_i = \sum_{x \in \Omega} \pi(x) \gamma_i(x). \tag{3.2.1}$$

Because in a saturated switch queues are always non-empty, γ_i can also be interpreted as the service rate at queue i (i.e., $\gamma_i = 1/\mathbb{E}[B_i]$).

At various points in this chapter, we will be interested in the throughput of a switch consisting of only a subset $I \subseteq \{1, \ldots, N\}$ of the queues. This $|I| \times M$ switch is constructed by removing all input queues j with $j \notin I$ from the original switch. The throughput of queue i in this reduced switch will be denoted by γ_i^I, with $i \in I$. If $i \notin I$, we define $\gamma_i^I = 0$.

3.3 Stability conditions

In this section we devise a heuristic that approximates the stability conditions for each queue in a non-saturated switch. Here, we define a queue to be stable if its throughput is equal to its arrival rate and unstable if its throughput is less than its arrival rate.

We use an idea of Ibe and Cheng [72] who devise a heuristic approach to determine the stability conditions of k-limited polling systems. The idea behind the heuristic is the following: Suppose that during a period $[-T, 0)$, work arrives to a server. After time 0, new arrivals are blocked and the work present is processed. Ibe and Cheng then state that a queue is stable if and only if on average its contents are processed in a period that is shorter than T time units, i.e., the queue is empty before time T. Chang and Lam [38] later proved rigorously that, for k-limited polling systems, this condition is indeed the true stability condition.

We apply the idea of Ibe and Cheng to an arbitrary $N \times M$ switch, but instead of letting packets arrive, we consider a deterministic fluid approximation where fluid enters (leaves) the buffer at the same rate at which packets would arrive (depart). At time 0 all queues of the switch will be occupied, leading to a saturated $N \times M$ switch of which the service rates can be computed. All queues are then drained at these rates until the first queue becomes empty. Since we block new arrivals, this queue remains empty from that moment on. The other queues still have work present, so they constitute a saturated $(N-1) \times M$ switch. We determine the service rates in this switch, drain the queues at these rates and compute the time at which the next queue becomes empty. We repeat this process until all queues are empty.

We approximate the stability condition of a queue by the condition that it is empty at time T. While Chang and Lam [38] proved that the heuristic indeed gives the right stability conditions for k-limited polling systems, we have to conclude (see Sections 3.6, 3.7, and 3.8) that the heuristic only yields an approximation for switches, albeit a very accurate one.

For clarity, we introduce notation before we describe the heuristic more precisely. The time period during which all queues are non-empty is called the first step, the time period during which all but one queues are non-empty the second step, and so on. We define $t^{(n)}$ as the time at which the nth step ends. We let $q_i^{(n)}$ denote the fluid level of queue i at the beginning of step n of the procedure, i.e., at time $t^{(n-1)}$. We further denote the service rate of queue i in step n by $\gamma_i^{(n)}$ and e_n as the index of the queue that is the nth to become empty. The set of non-empty queues in step n is denoted by $V^{(n)}$.

Algorithm 3.3.1.

Initialise $q_i^{(1)} = \lambda_i T$, $V^{(1)} = \{1, \dots, N\}$, and let $t^{(0)} = 0$. For $n = 1, \dots, N$:

- Calculate service rates $\gamma_i^{(n)} := \gamma_i^{V^{(n)}}$ using (3.2.1), where $\gamma_i^{V^{(n)}}$ is the steady-state throughput of the saturated $|V^{(n)}| \times M$ switch that is constructed by removing all queues $j \notin V^{(n)}$ from the original switch.

- The index of the nth queue to become empty is given by

$$e_n = \arg\min_{i \in V^{(n)}} \left\{ \frac{q_i^{(n)}}{\gamma_i^{(n)}} \right\}, \tag{3.3.1a}$$

since $q_i^{(n)}/\gamma_i^{(n)}$ is the time it would take to empty queue i if it was allowed to serve packets at rate $\gamma_i^{(n)}$ indefinitely. The time at which queue e_n becomes empty is given by

$$t^{(n)} = t^{(n-1)} + \frac{q_{e_n}^{(n)}}{\gamma_{e_n}^{(n)}} = t^{(n-1)} + \min_{i \in V^{(n)}} \left\{ \frac{q_i^{(n)}}{\gamma_i^{(n)}} \right\}. \tag{3.3.1b}$$

- Queue e_n is empty from now on, so

$$V^{(n+1)} = V^{(n)} \setminus \{e_n\}. \tag{3.3.1c}$$

The remaining fluid in the other queues is given by

$$q_i^{(n+1)} = q_i^{(n)} - (t^{(n)} - t^{(n-1)})\gamma_i^{(n)}, \qquad \text{for } i \in V^{(n+1)}. \tag{3.3.1d}$$

Remark 3.3.2. If there are multiple i (say $i = i_1, \ldots, i_k$) for which $q_i^{(n)}/\gamma_i^{(n)}$ is minimal, we can arbitrarily choose one (say i_1). For $i = i_2, \ldots, i_k$, $q_i^{(n+1)} = 0$, so that $t^{(n+k-1)} = t^{(n+k-2)} = \ldots = t^{(n)}$.

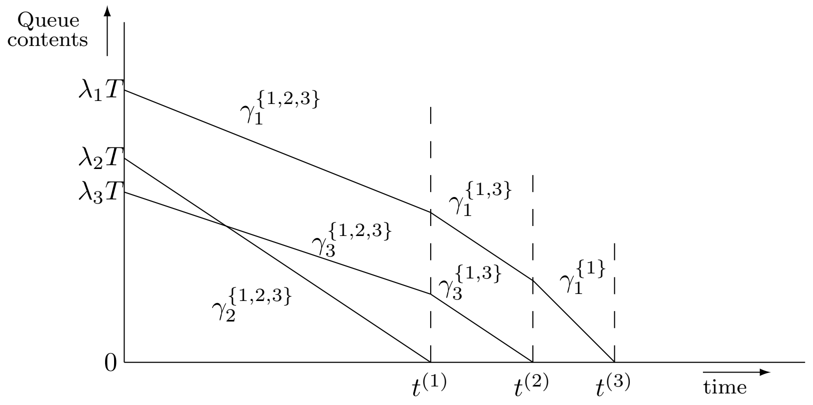


Figure 3.2: A schematic representation of the queue contents during Algorithm 3.3.1. All queues start with a fluid level of $\lambda_i T$ and are served at rate $\gamma_i^{(1)} = \gamma_i^{\{1,2,3\}}$ until the first queue becomes empty at time $t^{(1)}$. The first queue to become empty is queue 2 (so $e_1 = 2$), which is subsequently removed from the switch. The two remaining queues are served at rates $\gamma_i^{(2)} = \gamma_i^{\{1,3\}}$ for $i = 1, 3$, until $t^{(2)}$, which is when queue 3 becomes empty (so $e_2 = 3$). Finally, the last queue is served at rate $\gamma_1^{(3)} = \gamma_1^{\{1\}} = 1$ until time $t^{(3)}$.

Lemma 3.3.3. *Define $\Delta t^{(n)} = t^{(n)} - t^{(n-1)}$ for $n \geq 1$ (recall that $t^{(0)} = 0$). We can write $\Delta t^{(n)}$ as*

$$\Delta t^{(n)} = T \sum_{m=1}^{n} d_{m,n} \lambda_{e_m}, \tag{3.3.2}$$

where the constants $d_{m,n}$ are recursively determined by

$$d_{m,n} = \begin{cases} -\sum_{k=m}^{n-1} \frac{\gamma_{e_n}^{(k)}}{\gamma_{e_n}^{(n)}} d_{m,k}, & \text{for } m = 1, \ldots, n-1, \\ 1/\gamma_{e_n}^{(n)}, & \text{for } m = n. \end{cases}$$

Proof. We prove this statement by induction on n. From Eqs. (3.3.1a) and (3.3.1b) with $n = 1$ it follows that indeed $\Delta t^{(1)} = T\lambda_{e_1}/\gamma_{e_1}^{(1)}$. Suppose that the statement is true for $1, \ldots, n-1$. By recursively applying (3.3.1d) it follows that

$$q_{e_n}^{(n)} = \lambda_{e_n} T - \sum_{k=1}^{n-1} \gamma_{e_n}^{(k)} \Delta t^{(k)}.$$

We get

$$\begin{aligned} \Delta t^{(n)} = \frac{q_{e_n}^{(n)}}{\gamma_{e_n}^{(n)}} &= \frac{\lambda_{e_n} T}{\gamma_{e_n}^{(n)}} - \sum_{k=1}^{n-1} \frac{\gamma_{e_n}^{(k)}}{\gamma_{e_n}^{(n)}} \Delta t^{(k)} \\ &= T \frac{\lambda_{e_n}}{\gamma_{e_n}^{(n)}} - T \sum_{k=1}^{n-1} \sum_{m=1}^{k} \frac{\gamma_{e_n}^{(k)}}{\gamma_{e_n}^{(n)}} d_{m,k} \lambda_{e_m} \qquad \text{(induction hypothesis)} \\ &= T \frac{\lambda_{e_n}}{\gamma_{e_n}^{(n)}} - T \sum_{m=1}^{n-1} \lambda_{e_m} \sum_{k=m}^{n-1} \frac{\gamma_{e_n}^{(k)}}{\gamma_{e_n}^{(n)}} d_{m,k} = T \sum_{m=1}^{n} d_{m,n} \lambda_{e_m}. \end{aligned}$$

□

Corollary 3.3.4. *The time at which queue e_n becomes empty is given by:*

$$t^{(n)} = \sum_{k=1}^{n} \Delta t^{(k)} = T \sum_{k=1}^{n} \sum_{m=1}^{k} d_{m,k} \lambda_{e_m}. \tag{3.3.3}$$

Approximation 3.3.5. *We approximate the stability condition of queue e_n by $t^{(n)} < T$, or equivalently:*

$$\sum_{k=1}^{n} \sum_{m=1}^{k} d_{m,k} \lambda_{e_m} < 1. \tag{3.3.4}$$

If (3.3.4) is satisfied, we say that queue e_n is approximately stable, otherwise we say queue e_n is approximately unstable.

The stability condition derived in this section is an approximation due to three reasons: First, we look at a process in which we block the arrivals after time 0, while in reality new packets keep on arriving (and interfere with packets that arrived before time 0). Second, we look at a deterministic fluid process instead of a stochastic process in which packets arrive and depart. And third, we assume that if one of the queues becomes empty, the service rate of the other queues instantaneously becomes the steady-state service rate of a saturated switch with one input port less. A numerical evaluation of the accuracy of the Approximation 3.3.5 can be found in Sections 3.6, 3.7, and 3.8, where we compare the stability condition with results obtained from a simulation. It will be shown that the relative error of the approximation is generally less than 2%.

In the sequel, we assume that the arrival rates, or *loads*, are distributed over the queues according to a fixed weight vector $(\nu_1, \dots, \nu_N)$, with $\sum_i \nu_i = 1$, such that $\lambda_i = \nu_i \lambda$, where λ is the total load. It will be convenient to have a labelling of the queues such that queue 1 is the first to become approximately unstable as λ increases, queue 2 the second, and so on. We therefore make the following assumption:

Assumption 3.3.6. As λ increases, the queues become approximately unstable in increasing order, i.e.,

$$e_i = N + 1 - i. \tag{3.3.5}$$

The rationale behind Equation (3.3.5) is that the queue that needs the most time to process the fluid (i.e., queue e_N, the *last* queue to become empty) is the *first* queue to become unstable if the load λ is increased. Likewise, the *second-last* queue to become empty (i.e., queue e_{N-1}) is the *second* to become unstable, and so on. Assumption 3.3.6 is not restrictive in any way; it can be trivially obtained by a simple reordering of the queues.

Because $\lambda_{e_n} = \nu_{e_n}\lambda$ and $i = e_{N-i+1}$, the approximate stability condition of queue i (see (3.3.4)) reduces to

$$\lambda \sum_{k=1}^{N-i+1} \sum_{m=1}^{k} d_{m,k} \nu_{N-m+1} < 1. \tag{3.3.6}$$

We define $\lambda_{sat,i}$ such that queue i is approximately stable if $\lambda < \lambda_{sat,i}$ and approximately unstable otherwise. It follows from (3.3.6) that

$$\lambda_{sat,i} = \left(\sum_{k=1}^{N-i+1} \sum_{m=1}^{k} d_{m,k} \nu_{N-m+1} \right)^{-1}. \tag{3.3.7}$$

3.4 Throughput

In this section, we study the throughput of each queue of the switch as a function of a single load parameter λ. To illustrate how the throughput depends on λ, we show the simulation throughput per queue in Figure 3.3 for an example. This example will be used as a running example throughout this and the next section, as the characteristics displayed in this plot are typical for non-uniform switches. In the example

$$P = (p_{ij}) = \begin{pmatrix} 0.1 & 0.3 & 0.4 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix},$$

and ν is given by

$$\nu = \begin{pmatrix} 0.35 & 0.3 & 0.2 & 0.15 \end{pmatrix}.$$

The arrivals in this example are governed by Bernoulli processes with parameter $\min\{1, \lambda\nu_i\}$, i.e., each time slot an arrival takes place at queue i with probability $\min\{1, \lambda\nu_i\}$. Because Bernoulli processes are not defined for parameters greater than 1, we have to limit the parameter to 1 if λ grows large. This does not have any influence since at most one packet per queue may be served each time slot anyway. The queue is thus saturated and the steady state behaviour of saturated queues does not change if the load is further increased.

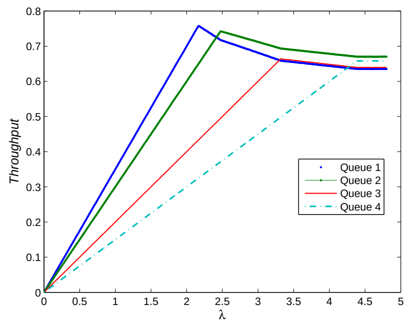


Figure 3.3: A typical plot of the throughput of the queues.

For each queue the throughput initially increases linearly in λ because the throughput is equal to the arrival rate $\lambda_i = \lambda\nu_i$. The throughput of each queue reaches its peak at the load for which that queue becomes unstable. Beyond this load the throughput decreases, roughly linearly, due to the fact that the load at other, still stable, queues increases (and as a consequence also the throughput of these queues increases). It does so until the next queue becomes unstable, after which it keeps decreasing roughly linearly, but with a different slope. Finally, for loads that are high enough, the throughput is constant. In this case, the load is so high that the switch has become saturated.

We are particularly interested in the region where the throughput is decreasing, which corresponds to the situation where some queues are stable while others are not. It turns out that the throughput in this region can be approximated by the average amount of fluid that has disappeared during $[0, T)$ in the process described in Algorithm 3.3.1. We will describe this approximation in more detail now.

We define l_λ as the number of approximately stable queues:

$$l_\lambda := |\{i : \lambda < \lambda_{sat,i}\}| = |\{n : t^{(n)} < T\}|. \tag{3.4.1}$$

The *total* amount of fluid that is drained from queue i in $[0,T)$ during the process described in Algorithm 3.3.1 is now given by

$$\sum_{n=1}^{l_\lambda} \gamma_i^{(n)} \Delta t^{(n)} + (T - t^{(l_\lambda)})\gamma_i^{(l_\lambda+1)}. \tag{3.4.2}$$

After all, queue i is drained at rate $\gamma_i^{(n)}$ in $[t^{(n-1)}, t^{(n)})$, as long as $t^{(n)} < T$. After time $t^{(l_\lambda)}$, the queue is drained at rate $\gamma_i^{(l_\lambda+1)}$ until time T.

We use the *average* rate at which fluid is drained as an approximation of the throughput of queue i, $\phi_i(\lambda)$. Dividing (3.4.2) by T yields:

$$\begin{aligned}\phi_i(\lambda) &= \frac{1}{T}\left[\sum_{n=1}^{l_\lambda} \gamma_i^{(n)} \Delta t^{(n)} + (T - t^{(l_\lambda)})\gamma_i^{(l_\lambda+1)}\right] \\ &= \frac{1}{T}\left[\sum_{n=1}^{l_\lambda} \gamma_i^{(n)} \Delta t^{(n)} - \gamma_i^{(l_\lambda+1)} \sum_{n=1}^{l_\lambda} \Delta t^{(n)}\right] + \gamma_i^{(l_\lambda+1)} \\ &= \lambda \sum_{n=1}^{l_\lambda} (\gamma_i^{(n)} - \gamma_i^{(l_\lambda+1)}) \sum_{m=1}^{n} d_{m,n}\nu_{N-m+1} + \gamma_i^{(l_\lambda+1)}\end{aligned}$$

by Eq. (3.3.2). Furthermore, for all i and k, $\gamma_i^{(k)} = \gamma_i^{V^{(k)}}$, where $V^{(k)} = \{1,\dots,N\} \setminus \{e_1,\dots,e_{k-1}\} = \{1,\dots,N-k+1\}$. As a result:

$$\phi_i(\lambda) = \lambda \sum_{n=1}^{l_\lambda} \left(\gamma_i^{\{1,\dots,N-n+1\}} - \gamma_i^{\{1,\dots,N-l_\lambda+1\}}\right) \sum_{m=1}^{n} d_{m,n}\nu_m + \gamma_i^{\{1,\dots,N-l_\lambda\}}. \tag{3.4.4}$$

Note that $\phi_i(\lambda)$ is a piecewise linear function. Its slope changes at $\lambda = \lambda_{sat,k}$, $k = 1,\dots,N$, since those are the values for which l_λ changes.

A number of interesting observations can be made about $\phi_i(\lambda)$. First, because stable queues are empty at time T, the *total* amount of fluid processed in $[0,T)$ from a stable queue i is equal to $\lambda\nu_i T$. It thus follows that

$$\phi_i(\lambda) = \lambda\nu_i, \qquad \text{for } \lambda \le \lambda_{sat,i},$$

i.e., the throughput approximation of a stable queue is its arrival rate. Second, if $\lambda \ge \lambda_{sat,N}$ then all queues are approximately unstable and hence $l_\lambda = 0$. In this case Eq. (3.4.4) directly implies that

$$\phi_i(\lambda) = \gamma_i^{\{1,\dots,N\}}, \qquad \text{for } \lambda \ge \lambda_{sat,N},$$

i.e., the throughput approximation is equal to the throughput under saturation. Third, with a load λ there are l_λ stable queues, so queues $1,\dots,N-l_\lambda$ are unstable. For unstable queues i, i.e., for $i \le N - l_\lambda$, the constant term in Eq. (3.4.4), $\gamma_i^{\{1,\dots,N-l_\lambda\}}$, is the service rate of queue i in a saturated switch consisting of only queues $1, \ldots, N - l_\lambda$. In other words, given a certain load, the constant term in the throughput approximation of an unstable queue is given by its service rate in a saturated switch consisting of only the unstable queues.

The following lemma is stated for future reference:

Lemma 3.4.1. *Suppose that i is such that $\lambda_{sat,i} \leq \lambda < \lambda_{sat,i+1}$, with $\lambda_{sat,N+1} := \infty$. The throughput approximation of queue i can now be rewritten in the following form:*

$$\phi_i(\lambda) = \gamma_i^{\{1,\ldots,i\}} + \lambda \left(\nu_i - \frac{\gamma_i^{\{1,\ldots,i\}}}{\lambda_{sat,i}} \right). \tag{3.4.5}$$

Proof. From (3.4.1) we obtain $i = N - l_\lambda$. From (3.4.4) it thus follows that $\phi_i(\lambda) = c_i\lambda + \gamma_i^{\{1,\ldots,i\}}$ for some constant c_i and $\lambda_{sat,i} \leq \lambda < \lambda_{sat,i+1}$. In particular this expression holds for $\lambda = \lambda_{sat,i}$, which implies $\phi_i(\lambda_{sat,i}) = c_i\lambda_{sat,i} + \gamma_i^{\{1,\ldots,i\}}$. Furthermore, $\phi_i(\lambda_{sat,i}) = \nu_i\lambda_{sat,i}$. Equating the latter two expressions, solving with respect to c_i, and rewriting yields (3.4.5). $\square$

A schematic plot of ϕ_i for a 3×3 switch can be found in Figure 3.4. Clearly, it has roughly the same characteristics as Figure 3.3. A more detailed analysis of the accuracy of ϕ_i is conducted in Sections 3.6, 3.7, and 3.8.

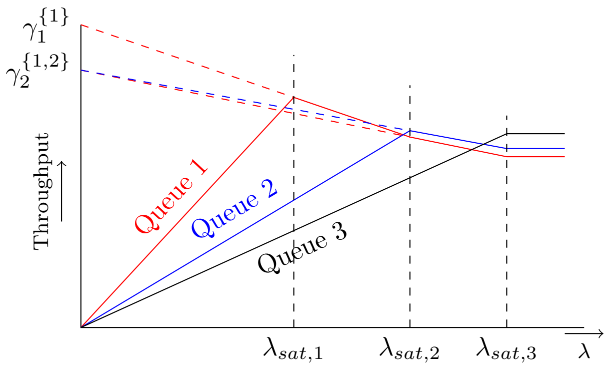


Figure 3.4: A schematic representation of the throughput approximation $\phi_i(\lambda)$ in a 3×3 switch. If queue i is approximately stable, the throughput approximation is given by the arrival rate. If queue i is approximately unstable the throughput approximation has a y-intercept given by the service rate of queue i in a saturated switch consisting of only the unstable queues. It can easily be argued that $\gamma_1^{\{1\}} = 1$ and $\gamma_1^{\{1,2\}} = \gamma_2^{\{1,2\}}$ (in a saturated $2 \times M$ switch, the throughput is $1 - p/2$ for both queues, where p is the probability that there is contention in an arbitrary time slot).

3.5 Waiting time and service rate

The main goal of this section is to present an approximation of the mean waiting time of packets in queue i (Section 3.5.1). A key ingredient in this mean waiting time approximation is an approximation of the service rate of queue i, which is derived in Section 3.5.2. Recall that the service time of an arbitrary packet in queue i, denoted by B_i, is defined as the time spent by that packet in the head position of queue i. The service rate of queue i is defined as $1/\mathbb{E}[B_i]$.

3.5.1 Waiting time

The key assumption of our mean waiting time approximation is that B_i follows a geometric distribution, similar to the mean waiting time approximation for uniform switches in Chapter 2. The rationale behind this approximation is as follows: If there are k packets with the same destination, each one is selected with probability $1/k$, without taking previous time slots into account. The service time is equal to the number of attempts to be served (including the successful one), where each attempt is successful with a *different* probability, as the number of packets with the same destination varies over time. We approximate the service times by neglecting this difference; the geometric distribution describes the number of attempts where each attempt is successful with the *same* probability.

We will derive a service rate approximation $\mu_i(\lambda) \approx 1/\mathbb{E}[B_i]$, which is used as the success probability of the approximating geometric distribution. We then approximate the waiting time of a packet in queue i by the waiting time in a $./Geo/1$ queue. The arrival process to the $./Geo/1$ queue is the same as the arrival process to the switch, i.e., for a switch with Bernoulli arrivals (and hence geometric interarrival times), the approximating queue is a $Geo/Geo/1$ queue, with a Markovian Arrival Process (MAP), it is a $MAP/Geo/1$ queue, and so on. For most arrival processes, the mean waiting time approximation follows from known results. For instance, with Bernoulli arrivals, the mean waiting time approximation is the following (see, e.g., Equation (1.19a) of Takagi [134], or Equation (2.1.1) of this manuscript):

$$\mathbb{E}[W_i] \approx \frac{\lambda\nu_i(1-\mu_i(\lambda))}{\mu_i(\lambda)(\mu_i(\lambda)-\lambda\nu_i)}, \qquad \text{for } \lambda\nu_i < \mu_i(\lambda), \tag{3.5.1}$$

where W_i is the waiting time of a packet in queue i. For a $MAP/Geo/1$ queue, the mean waiting time can be determined numerically using the theory of quasi birth death processes (see, e.g., Neuts [107]).

3.5.2 Service rate

In this subsection, we derive our service rate approximation $\mu_i(\lambda)$. Before giving the approximation, we first show some simulation results of $1/\mathbb{E}[B_i]$ in Figure 3.5 in order to understand the behaviour of the service rate better. Here, $\mathbb{E}[B_i]$ was obtained by taking the average of measurements of the service times of individual packets. The input of this figure is again the 'running example' (see p. 48).

First, up to the first saturation load, the service rate shows a certain curvature. As in Chapter 2, we propose a quadratic interpolation for this part. Second, the service rates appear to be piecewise linear between the various saturation loads. We propose a linear approximation for these parts. Finally, we observe that if all queues

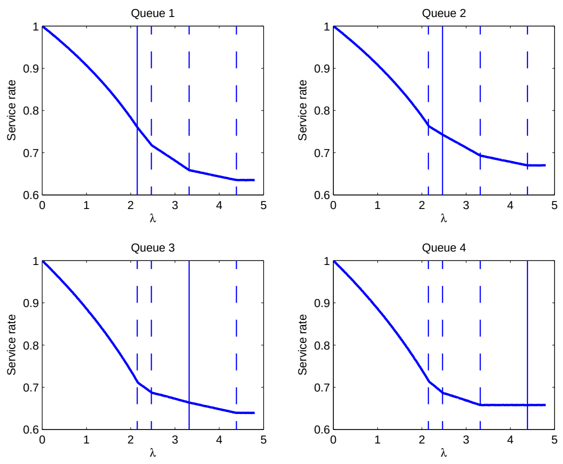


Figure 3.5: The service rates of all queues of the 'running example'. The vertical lines correspond to the saturation loads. In the figure of queue i, the saturation load of queue i itself is indicated by the solid line, while the saturation loads of the other queues are indicated by dashed lines.

are saturated, the service rates are equal to the service rates in a saturated switch.

It remains to approximate the service rates at saturation loads and the parameters of the quadratic interpolation; once we have approximated the service rates at saturation loads, the linear interpolations follow. The proposed service rate approximation can thus be summarised as follows:

$$\mu_i(\lambda) = \begin{cases} a_i + b_i\lambda + c_i\lambda^2, & \text{for } \lambda < \lambda_{sat,1}, \\ \mu_i(\lambda_{sat,k}) + \frac{\mu_i(\lambda_{sat,k+1})-\mu_i(\lambda_{sat,k})}{\lambda_{sat,k+1}-\lambda_{sat,k}}(\lambda - \lambda_{sat,k}), & \text{for } \lambda_{sat,k} \le \lambda < \lambda_{sat,k+1}, \\ \gamma_i, & \text{for } \lambda \ge \lambda_{sat,N}, \end{cases}$$

where the $\mu_i(\lambda_{sat,k})$, $i, k = 1, \dots, N$ are the approximations of the service rates at saturation loads, and a_i, b_i, and c_i the parameters of the quadratic interpolation.

The service rates at saturation loads will be expressed in terms of the throughput in saturated switches, the saturation load, and the throughput of an unstable queue given that other queues are stable. The values of the parameters a_i, b_i, and c_i will be expressed in terms of elements from the P-matrix and the ν-vector.

Derivation of $\mu_i(\lambda_{sat,k})$

We propose an algorithm that can be used to find the service rate approximation at saturation loads, $\mu_i(\lambda_{sat,k})$, for $i = 1, \dots, N$, $k = 1, \dots, N$. We first present the approximation in a concise form, and then we explain the underlying equations.

Approximation 3.5.1. *For all k:*

If $i \le k$,

$$\mu_i(\lambda_{sat,k}) = \phi_i(\lambda_{sat,k}). \tag{3.5.2}$$

If $i = k+1$,

$$\mu_i(\lambda_{sat,k}) = \mu_i(\lambda_{sat,i-1}) = \gamma_i^{\{1,\ldots,i\}} + \lambda_{sat,i-1}\left(\nu_i - \frac{\gamma_i^{\{1,\ldots,i\}}}{\lambda_{sat,i}}\right). \tag{3.5.3}$$

If $i \geq k+2$, $\mu_i(\lambda_{sat,k}) = 1/b_i$, with b_i the solution of the following set of equations:

$$b_i = \sum_{J:i\in J} \frac{1}{\gamma_i^J} \prod_{\substack{j\in J\\ j\neq i}} \rho_j \prod_{j\notin J}(1-\rho_j), \qquad i \geq k+2, \tag{3.5.4}$$

where

$$\rho_j = \begin{cases} 1 & \text{for } j \leq k, \\ \nu_j\lambda_{sat,k}/\mu_j(\lambda_{sat,k}), & \text{for } j = k+1, \\ \nu_j\lambda_{sat,k}b_j, & \text{for } j \geq k+2, \end{cases}$$

with $\mu_j(\lambda_{sat,k})$ for $j = k+1$ as in (3.5.3).

Remark 3.5.2. Equation (3.5.4) expresses b_i in terms of b_j, $j \neq i$, and $i, j \geq k+2$, which indeed yields a set of equations in $b_{k+2}, \ldots, b_N$. The solution to this set of equations is the mean service time approximation; the service rate approximation follows by setting $\mu_j(\lambda_{sat,k}) = 1/b_j$. For general N we cannot guarantee that a unique solution in $[1, N]$ of (3.5.4) exists. However, in all our numerical examples we found precisely one solution in $[1, N]$. Here, the maximal value is N since in the worst case N packets have the same destination and one of them is selected for service at random.

Explanation of Equation (3.5.2)
The rationale behind Equation (3.5.2) is that the service rate of an unstable queue is given by its throughput. Because the queues become approximately unstable in increasing order (see Assumption 3.3.6) we know that, when $\lambda = \lambda_{sat,k}$, queue i is unstable if $i \leq k$, so $\mu_i(\lambda_{sat,k}) = \phi_i(\lambda_{sat,k})$ if $i \leq k$.

Explanation of Equation (3.5.3)
For $i = k+1$, we use an interesting observation from Figure 3.5: When a queue reaches its saturation load, the slope of the service rate does not change, i.e., the service rate keeps changing in the same way at any of the solid vertical lines.

This phenomenon can be explained intuitively by the following reasoning: We are interested in the service rate of queue i, which means that we look at the switch conditioned on the event that queue i is non-empty. By increasing the load, the other stable queues take away capacity from queue i which leads to a lower service rate of queue i. Since the load of the queues is increased linearly, the capacity taken away from queue i by the other queues also grows linearly. Because we look at the switch conditioned on the event that queue i is non-empty, the rate at which the service rate of queue i decreases does not change when queue i saturates; it only changes when one of the *other* queues becomes unstable.

The service rate approximation we propose takes this phenomenon into account. By Equation (3.5.2), the slope of $\mu_i(\lambda)$ for $\lambda \in [\lambda_{sat,i}, \lambda_{sat,i+1})$ is known. Equation (3.5.3) now follows from taking the slope of $\mu_i(\lambda)$ for $\lambda \in [\lambda_{sat,i-1}, \lambda_{sat,i})$ the same as that of $\mu_i(\lambda)$ for $\lambda \in [\lambda_{sat,i}, \lambda_{sat,i+1})$ (see also Lemma 3.4.1 and use $\mu_i(\lambda) = \phi_i(\lambda)$ for all $\lambda \geq \lambda_{sat,i}$).

Explanation of Equation (3.5.4)
In (3.5.4), ρ_j is an approximation for the probability that queue j is non-empty. Unstable queues are always non-empty, so $\rho_j = 1$ for $j \leq k$. Stable queues are non-empty with a probability equal to the arrival rate multiplied by the mean service time. For queue $j = k+1$, we approximate the mean service time by $1/\mu_j(\lambda_{sat,k})$ (with $\mu_j(\lambda_{sat,k})$ as in Equation (3.5.3)), so $\rho_j = \nu_j \lambda_{sat,k}/\mu_j(\lambda_{sat,k})$. For queues $j \geq k+2$ we use the unknown b_j as the mean service time approximation, so $\rho_j = \nu_j \lambda_{sat,k} b_j$.

Next, assuming independence, $\prod_{\substack{j \in J \\ j \neq i}} \rho_j \prod_{j \notin J} (1-\rho_j)$ is an approximation for the probability that, given that queue i is non-empty, all queues $j \in J$ are non-empty and all queues $j \notin J$ are empty. Finally, given that the queues in J are the only non-empty ones, we approximate the mean service time of queue i by $1/\gamma_i^J$.

Derivation of a_i, b_i, and c_i

As in Chapter 2, we study the behaviour of the switch in light traffic, i.e., we assume λ is small and neglect $\mathcal{O}(\lambda^2)$ terms, in order to obtain the parameters a_i, b_i, and c_i. The light traffic approximation typically depends on the specific arrival process assumed. Here we assume that arrivals are governed by Bernoulli processes with parameter $\min\{1, \lambda\nu_i\}$. The analysis outlined below shows the typical line of reasoning and can be extended to other arrival processes.

In light traffic, the service time of an arbitrary (tagged) packet in queue i is given by

$$B_i = \begin{cases} 1 & \text{w.p. } 1 - \frac{1}{2}\beta_i\lambda + \mathcal{O}(\lambda^2), \\ 2 & \text{w.p. } \frac{1}{2}\beta_i\lambda + \mathcal{O}(\lambda^2), \\ k > 2 & \text{w.p. } \mathcal{O}(\lambda^2), \end{cases} \tag{3.5.5}$$

and consequently

$$\frac{1}{\mathbb{E}[B_i]} = 1 - \frac{1}{2}\beta_i\lambda + \mathcal{O}(\lambda^2),$$

where

$$\beta_i = \sum_{k \neq i} \nu_k \sum_{j=1}^{N} p_{i,j} p_{k,j}.$$

This follows from the fact that $\beta_i\lambda + \mathcal{O}(\lambda^2)$ is the probability that another packet arrives in the same time slot with the same destination as our tagged packet, and with probability 1/2 the tagged packet loses the contention. Furthermore, events in which, next to our tagged packet, two or more other packets are involved all occur with probability $\mathcal{O}(\lambda^2)$ because they require the simultaneous arrival of two or more other packets at some point in time.

We choose the parameters a_i and b_i such that the service rate approximation captures this light traffic behaviour. Furthermore, the parameter c_i is chosen such that the service rate approximation is continuous at $\lambda_{sat,1}$. For $\lambda < \lambda_{sat,1}$, we thus have the following quadratic approximation:

$$\mu_i(\lambda) = 1 - \frac{1}{2}\beta_i\lambda + \frac{-1 + \frac{1}{2}\beta_i\lambda_{sat,1} + \mu_i(\lambda_{sat,1})}{\lambda_{sat,1}^2}\lambda^2. \tag{3.5.6}$$

Remark 3.5.3. The waiting time approximation depends in two ways on the arrival process to the switch: First, the waiting time in, for instance, a $Geo/Geo/1$ queue is different from that in a $MAP/Geo/1$ queue. Even with the same service rate, the mean waiting time approximations are thus still different. Second, the service rate approximation depends on the specific arrival process via the light traffic analysis.

Remark 3.5.4. Computing the service rate approximation requires the computation of the saturation throughput of all 2^N possible subswitches (see Eq. (3.5.4)). Clearly, this can be problematic if N is large, but for switches in networks on chips N is typically small.

3.6 Analysis of the running example

In this and the next two sections we illustrate the accuracy of the approximations devised in Section 3.3, 3.4, and 3.5, by a comparison with simulation. The example we study in detail in this section is the running example, combined with Bernoulli arrivals. In Section 3.7, we analyse this example combined with correlated traffic models. In Section 3.8, we conduct a larger numerical study of 100 examples with Bernoulli arrivals, where we draw more general conclusions.

Recall that the arrivals are governed by independent Bernoulli processes with parameter $\min\{\nu_i\lambda, 1\}$ and that for the running example, P is as follows:

$$P = (p_{ij}) = \begin{pmatrix} 0.1 & 0.3 & 0.4 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}.$$

Furthermore, the vector ν is given by

$$\nu = \begin{pmatrix} 0.35 & 0.3 & 0.2 & 0.15 \end{pmatrix}.$$

This section is divided into two parts: In 3.6.1 we study the accuracy of the throughput and the stability condition approximations, and in 3.6.2 that of the service rate and mean waiting time approximations.

3.6.1 Throughput and stability conditions

Recall that we defined a queue to be stable if its throughput is equal to its arrival rate. Because of this definition, we can look at the simulation throughput and if it deviates from the arrival rate, the queue is unstable. Recall furthermore that the value of λ for which a queue becomes unstable is called the 'saturation load' of that queue. During the entire numerical analysis, we make a distinction between the approximated saturation load of Approximation 3.3.5 and the saturation load observed via simulation (called the *observed* saturation load).

Each simulation run consists of 10^7 time slots, and measurements of the first 10^5 time slots were discarded. Each run was repeated for $n = 10$ times, after which λ was increased by 0.01. In Table 3.1 we give the simulation throughput of queue 1, for loads close to its saturation load. The fourth column of this table shows the standard deviation s of the throughput and the fifth shows the deviation of the throughput from the arrival rate in terms of the standard deviation.

λ	Arr. rate ($\nu_1\lambda$)	Avg. throughput($\overline{\phi}$)	St.dev.(s)	$\sqrt{n}(\nu_1\lambda - \overline{\phi})/s$
2.13	0.7455	0.7455	$1.4 \cdot 10^{-4}$	-0.28
2.14	0.7490	0.7489	$1.1 \cdot 10^{-4}$	1.49
2.15	0.7525	0.7525	$0.8 \cdot 10^{-4}$	1.58
2.16	0.7560	0.7560	$1.8 \cdot 10^{-4}$	-0.73
2.17	0.7595	0.7574	$2.0 \cdot 10^{-4}$	32.73
2.18	0.7630	0.7560	$1.6 \cdot 10^{-4}$	140.88
2.19	0.7665	0.7548	$0.9 \cdot 10^{-4}$	406.29

Table 3.1: Simulation throughput

In the last column it can be seen that up to a load of 2.16 the simulated throughput is within $1.6s/\sqrt{n}$ of the arrival rate. For a load of 2.17 the deviation of the throughput from the arrival rate is clearly significant ($32.73s/\sqrt{n}$), which means that the queue is unstable. We call $\lambda = 2.17$ the saturation load, even though the queue is already unstable for this load, which implies that the true saturation load lies somewhere between 2.16 and 2.17.

An overview of the observed saturation loads can be found in Table 3.2. The approximation is more accurate for queues that become unstable for higher loads. Nevertheless, in all cases, the relative error is limited to 1%. So although the stability condition found is a little off in some cases, it gives a very accurate approximation of the true stability condition, especially for queues 3 and 4. We will come back to this issue in Section 3.8.

Now that we have investigated the accuracy of the stability condition approximation, we move on to the analysis of the throughput approximation. In Section 3.4 we saw that the throughput approximation is piecewise linear in λ between the approximate saturation loads. It thus makes sense to look at the throughput approximation in these points, and compare them with simulation values. In order to achieve a fair comparison, we compare the values of the throughput approximation

	Saturation load	
	Simulation	Alg. 3.3.1
Queue 1	2.17	2.1470
Queue 2	2.48	2.4669
Queue 3	3.33	3.3199
Queue 4	4.39	4.3869

Table 3.2: An overview of the stability conditions found.

in the approximated saturation loads to the simulation throughput in the observed saturation loads, even though these loads are not the same. For example, we compare the throughput approximation of queue 1 with load 2.1470 to the simulated throughput with load 2.17 rather than 2.1470.

The comparison of the throughput approximation and the simulated throughput can be found in Table 3.3. The diagonal of the table corresponds to the throughput of the queues at their saturation loads. The lower left triangle corresponds to the throughput of the queues if they are already unstable and another queue becomes so too. The upper right triangle of the table would correspond to the throughput of stable queues. However, in Section 3.4 it was mentioned that the throughput approximation is exact for stable queues. As a result, it would make no sense to compare the throughput approximation to simulation for different loads. The simulations for stable queues have therefore been omitted from the table.

Again we conclude from the table that the throughput approximation is in general very accurate, especially if more queues are unstable. In all cases the relative error is limited to 1%. The standard deviation of the throughput values is roughly between $1 \cdot 10^{-4}$ and $2 \cdot 10^{-4}$.

		Throughput			
	λ	Queue 1	Queue 2	Queue 3	Queue 4
Appr.	2.1470	0.7515			
Sim.	2.17	0.7571			
Appr.	2.4669	0.7144	0.7401		
Sim.	2.48	0.7170	0.7419		
Appr.	3.3199	0.6588	0.6933	0.6640	
Sim.	3.33	0.6586	0.6931	0.6638	
Appr.	4.3869	0.6532	0.6700	0.6395	0.6580
Sim.	4.39	0.6354	0.6700	0.6394	0.6580

Table 3.3: The accuracy of the throughput approximation.

3.6.2 Service rate and mean waiting time

In this subsection we study the accuracy of the service rate and mean waiting time approximations. In Figure 3.6 the service rate approximation is plotted together with simulation values. It is clear that our service rate approximation is

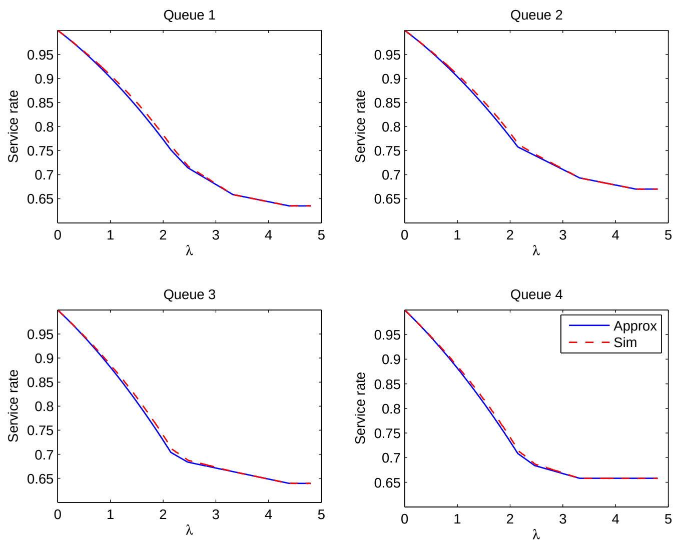


Figure 3.6: The service rate approximation.

indeed very accurate.

Even though the service rate approximation is accurate, our mean waiting time approximation is not necessarily accurate too. After all, an additional error might be induced by the assumption that the service time is geometrically distributed. A plot of the mean waiting time, together with its approximation, can be found in Figure 3.7. The figure shows that our approximation is quite accurate in this case, especially for queues 1 and 2.

We have plotted the relative error of our mean waiting time approximation in Figure 3.8. This figure reveals that the relative errors of queue 1 are generally within 5%, those of queue 2 and 3 generally within 10%, and the relative error of queue 4 takes values up to roughly 15%. Note that if λ is close to 0, the relative error varies greatly. This is caused by the fact that we divide two numbers close to 0.

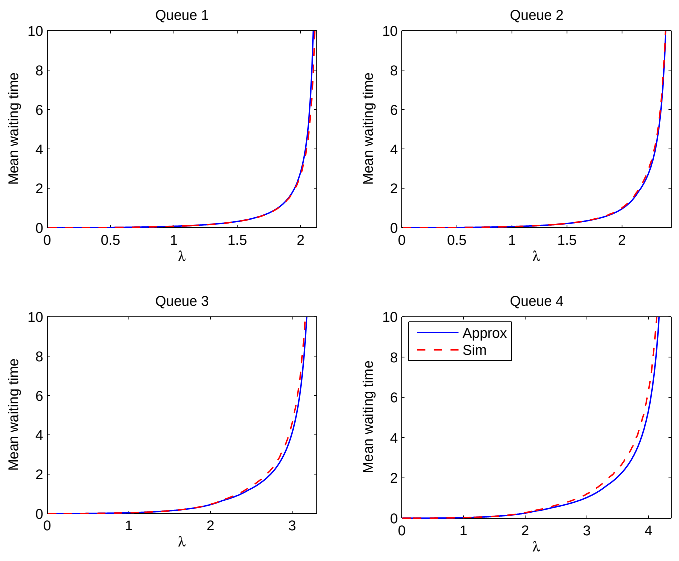


Figure 3.7: The mean waiting time approximation.

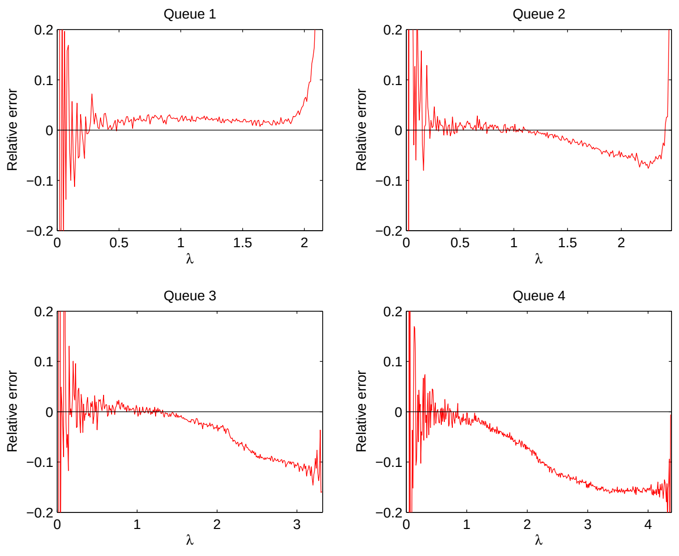


Figure 3.8: The relative error of the mean waiting time approximation

3.7 Correlated traffic

In this section we describe how our approximations can be applied to models with correlated traffic. We distinguish two types of correlated traffic: Correlation between arrivals (if an arrival takes place, it is more likely that the next time slot another arrival takes place as well), and correlation between destinations (if a packet has destination j, the next packet is more likely to have destination j as well).

3.7.1 Correlated arrivals

In this subsection, we analyse the running example where the arrival process to queue i is given by a 2-state MMBP (Markov Modulated Bernoulli Process, a special case of a MAP). The Markov chain $X_i(t)$ underlying the MMBP of queue i has state space $\{0,1\}$, and an arrival takes place at time t if and only if $X_i(t) = 1$. The transition probability matrix of $X_i(t)$ is given by

$$\begin{pmatrix} 1-\alpha_i & \alpha_i \\ 1-2\alpha_i & 2\alpha_i \end{pmatrix}.$$

In order to reduce the number of parameters, we assumed here that if an arrival takes place at time t, it is twice as likely that another arrival takes place at time $t+1$ than it would have been if no arrival had taken place at time t.

The arrival rate of queue i is given by

$$\lambda_i = \mathbb{P}(X_i(t) = 1) = \frac{\alpha_i}{1-\alpha_i}. \tag{3.7.1}$$

Given $\lambda_i = \nu_i\lambda$, the parameter α_i can thus be calculated.

Because the throughput in saturation is independent of the specific arrival process, the throughput and stability condition approximation of this example are the same as that of a switch with Bernoulli arrivals. The observed saturation load and throughput are also the same as that for a switch with Bernoulli arrivals (except for statistically insignificant deviations). We thus have good approximations for the saturation loads and throughput.

Since in an arbitrary time slot an arrival occurs at queue i with probability $\lambda_i = \nu_i\lambda$, the light traffic analysis for the MMBP is the same as that for Bernoulli arrivals, and hence the service rate approximation is the same too. The mean waiting time can now be approximated by that in an $MMBP/Geo/1$ queue. The mean waiting time can be determined numerically since the process $(X_i(t), Y_i(t))$, where $Y_i(t)$ is the length of the $MMBP/Geo/1$ queue, is a Markov chain on $\{0,1\}\times\mathbb{N}$. The equilibrium distribution of this Markov chain can be determined using the theory of quasi birth death (QBD) processes [91, 107].

Figure 3.9 displays the mean waiting time in the switch with an MMBP, its approximation, the mean waiting time in the switch with Bernoulli arrivals, and its approximation. The $MMBP/Geo/1$ approximation accurately predicts the higher waiting time induced by the correlation in the arrival process. The curves belonging

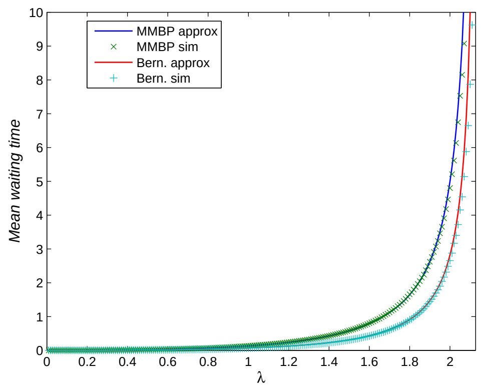


Figure 3.9: A comparison of the $MMBP/Geo/1$ and $Geo/Geo/1$ approximations.

to the Bernoulli arrival process are clearly different from those belonging to the MMBP.

3.7.2 Correlated destinations

In this subsection we consider a traffic model where packets consist of multiple flits, where a flit is precisely the amount of data that can be transmitted in one time slot. Instead of assuming that all flits arrive simultaneously, we assume that all flits in the packet arrive in consecutive time slots (*train* arrivals). The first flit of a packet is called a header.

During a geometrically distributed number of time slots (with expectation a), flits belonging to the same packet arrive to the switch. All flits in the packet have the same destination. After the complete arrival of the packet, there is an 'idle period' of a length that is geometrically distributed with parameter q_i. The length of this idle period may be equal to 0, i.e., two independent packets may arrive immediately after each other.

The expected length of an idle period is given by $1/q_i - 1$. The arrival rate of flits to input i is equal to the fraction of time slots in which an arrival takes place, i.e.,

$$\lambda_i = \frac{a}{a + \frac{1}{q_i} - 1}.$$

Rewriting this yields

$$q_i = \frac{1}{1 + a\frac{1-\lambda_i}{\lambda_i}}.$$

Given λ, ν and a, the parameter q_i can thus be calculated.

We assume that our switch uses wormhole routing, which entails that once the header has been transmitted to its output port, that output port is reserved for all flits in the packet. Only after the entire packet has been transmitted, other flits again contend for that output port.

The saturation throughput of the switch with train arrivals and wormhole routing is different from that of switches with Bernoulli arrivals. Nevertheless, the Markov chain approach used to determine the saturation throughputs can be extended in a straightforward manner by changing the transition probabilities to take train arrivals into account, and by enlarging the state space to take wormhole routing into account. Given the new values for these throughputs, the stability and throughput approximations follow in exactly the same way as before. The results can be found in Table 3.4 for various values of a. The accuracy of the approximation is comparable to that for a switch with Bernoulli arrivals ($a = 1$).

		$a=1$	$a=2$	$a=3$	$a=5$	$a=10$
Queue 1	Sim.	2.17	2.05	2.02	2.00	1.98
	Approx.	2.147	2.034	2.003	1.980	1.963
Queue 2	Sim.	2.48	2.33	2.29	2.26	2.23
	Approx.	2.467	2.315	2.273	2.242	2.221
Queue 3	Sim.	3.33	3.11	3.06	3.02	2.99
	Approx.	3.320	3.108	3.051	3.009	2.979
Queue 4	Sim.	4.39	4.08	4.00	3.93	3.90
	Approx.	4.387	4.071	3.986	3.923	3.878

Table 3.4: Saturation loads.

For the mean waiting time approximation, we need to translate the switch to a suitable approximating queue. This, however, is not straightforward because the service times of headers and non-headers differ; only headers of a train have to contend, while non-header packets always have a service time equal to 1. The extension of the mean waiting time approximation to train arrivals is beyond the scope of this thesis. The main point of this subsection is to show that the stability and throughput approximations can be applied to more sophisticated arrival models, such as train arrivals, without much effort and without significant loss of accuracy.

3.8 Numerical analysis

In this section, we study the performance of our approximation for Bernoulli arrivals on a much larger scale. We introduce ten matrices $P = (p_{ij})$ and ten vectors ν, and study the performance of the approximation of the 100 possible combinations. Out of these ten matrices and vectors, five of each have been chosen and five have been generated randomly.

The following five matrices have been chosen: First,

$$P = \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix},$$

which is the matrix corresponding to a uniform switch. Second,

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

since it corresponds to the situation in which all traffic has the same destination. This could, for instance, occur if a single memory is shared among multiple processors. Third,

$$P = \begin{pmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \end{pmatrix},$$

which corresponds to the situation where two outputs are equally likely. Fourth, the matrix of the running example is included,

$$P = \begin{pmatrix} 0.1 & 0.3 & 0.4 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix},$$

and fifth,

$$P = \begin{pmatrix} 0.1 & 0.7 & 0.1 & 0.1 \\ 0.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.7 \\ 0.1 & 0.1 & 0.7 & 0.1 \end{pmatrix},$$

where most traffic (i.e., a 0.7 fraction) has its own destination, and occasionally it deviates from this destination.

The following five ν vectors were chosen manually:

$$\begin{aligned} \nu =& (0.25, 0.25, 0.25, 0.25), \\ \nu =& (0.4, 0.3, 0.2, 0.1), \\ \nu =& (0.6, 0.2, 0.1, 0.1), \\ \nu =& (0.7, 0.1, 0.1, 0.1), \\ \nu =& (0.8, 0.1, 0.05, 0.05). \end{aligned}$$

The first vector is that of a uniform switch. The other vectors were chosen such that the first queue receives an increasingly higher load.

The following matrices were generated randomly:

$$P = \begin{pmatrix} 0.15 & 0.32 & 0.24 & 0.29 \\ 0.27 & 0.24 & 0.13 & 0.36 \\ 0.05 & 0.38 & 0.42 & 0.15 \\ 0.40 & 0.14 & 0.31 & 0.15 \end{pmatrix},$$

$$P = \begin{pmatrix} 0.28 & 0.29 & 0.15 & 0.28 \\ 0.34 & 0.03 & 0.15 & 0.48 \\ 0.15 & 0.21 & 0.30 & 0.34 \\ 0.40 & 0.04 & 0.01 & 0.55 \end{pmatrix},$$

$$P = \begin{pmatrix} 0.20 & 0.15 & 0.34 & 0.31 \\ 0.25 & 0.50 & 0.14 & 0.11 \\ 0.09 & 0.33 & 0.20 & 0.38 \\ 0.24 & 0.16 & 0.36 & 0.24 \end{pmatrix},$$

$$P = \begin{pmatrix} 0.11 & 0.31 & 0.02 & 0.56 \\ 0.18 & 0.48 & 0.26 & 0.08 \\ 0.55 & 0.41 & 0.01 & 0.03 \\ 0.37 & 0.03 & 0.23 & 0.37 \end{pmatrix},$$

$$P = \begin{pmatrix} 0.26 & 0.25 & 0.30 & 0.19 \\ 0.01 & 0.05 & 0.62 & 0.32 \\ 0.02 & 0.59 & 0.14 & 0.25 \\ 0.10 & 0.32 & 0.23 & 0.35 \end{pmatrix},$$

and the following five vectors were generated randomly:

$$\begin{aligned} \nu =& (0.37, 0.30, 0.25, 0.08), \\ \nu =& (0.34, 0.24, 0.23, 0.19), \\ \nu =& (0.38, 0.32, 0.20, 0.10), \\ \nu =& (0.40, 0.28, 0.17, 0.15), \\ \nu =& (0.31, 0.29, 0.27, 0.13). \end{aligned}$$

Each simulation of these 100 possible combinations ran for 10^7 time slots, and measurements of the first 10^5 time slots were discarded. The load λ was increased in steps of 0.01, until all queues became unstable. After the simulation, we renumbered the queues such that queue 1 is the first queue to become unstable, queue 2 the second, etc. We will give an overview of the errors of the saturation load and mean waiting time approximation.

An overview of the absolute value of the relative errors made in the saturation load approximation can be found in Table 3.5, and a graph of the empirical cumulative distribution in Figure 3.10. The absolute value of the relative error is the largest for queue 1, which we also saw in the previous section. On average, the error made for this queue is 1%, while for queues 2, 3, and 4 the error is even smaller. For queue 1, 95% of our simulations gave results within 2.2% error, and for the other queues within 0.87%.

	Queue 1	Queue 2	Queue 3	Queue 4
Avg. |rel. error|	0.010	0.0037	0.0024	0.0022
90% error quantile	0.020	0.0068	0.0047	0.0040
95% error quantile	0.022	0.0087	0.0063	0.0046

Table 3.5: The saturation load approximation.

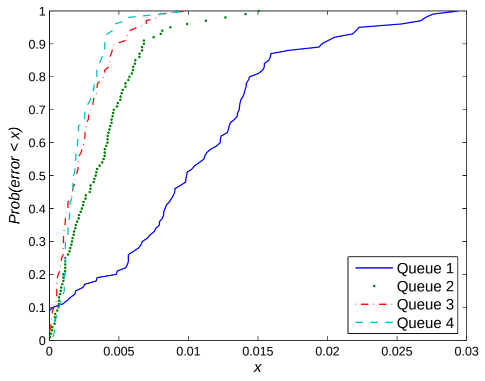


Figure 3.10: The empirical cumulative distribution of the errors of the saturation load approximation.

Note that even in cases where the approximation is exact, errors of up to 0.01 occur since this is the difference between consecutive simulation loads. For example, if the exact saturation load is 1.001, the observed saturation load would be either 1.00 or 1.01, depending on the precise values of p_{ij} and ν_i.

The approximation is more accurate for the queues that become unstable at higher loads. This can be understood if we recall that queue 4 is the first to become empty in Algorithm 3.3.1, queue 3 the second, and so on. It appears that we slightly underestimate the time at which the first queue becomes empty, the difference between this time and the time at which the second queue becomes empty, etc. All these errors combined entail that the saturation load approximation becomes less accurate for the queues that become unstable early on. In addition to this, we suspect that the algorithm becomes less accurate if N becomes larger.

We will now focus on the relative error (not its absolute value) of the mean waiting time approximation. Since each combination of a P-matrix and a ν-vector has its own set of saturation loads, we scale the loads in order to compare the various relative errors with each other. We do so by looking at $\rho_i := \lambda/\overline{\lambda}_{sat,i}$, where $\overline{\lambda}_{sat,i}$ is the observed saturation load. We then compare the relative error of the approximation for each value of ρ_i between 0 and 1. The mean of this error and the 5% and 95% quantile as functions of ρ_i are plotted in Figure 3.11.

Generally, the mean of the relative errors is reasonably close to 0, which indicates that our approximation is accurate on average. Furthermore, for ρ_i roughly up to 0.8, the error quantiles are well within -20% and 20%, except for queue 1. If ρ_i

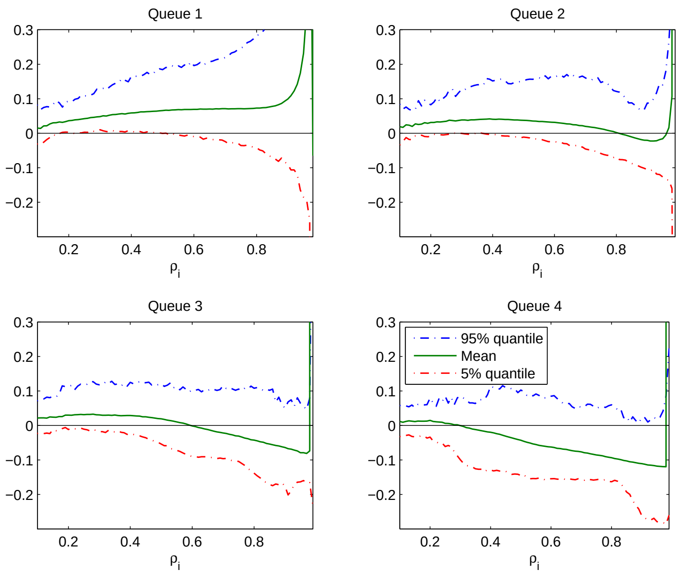


Figure 3.11: The relative error of the mean waiting time approximation. Note that the horizontal axis starts at $\rho_i = 0.1$. This has been done because for small ρ_i we have to divide two values close to 0, which is essentially meaningless.

approaches 1, our approximation clearly becomes less accurate and the difference between the error quantiles increases.

From the data we inferred that for the matrix

$$(p_{ij}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

the mean waiting approximation is the least accurate. An explanation for this phenomenon is that our service rate approximation is based on an independence assumption for the queues. With the matrix mentioned above, there is only one output port that is shared by all queues, which means that we have strong dependence between all queues. Of the ten ν vectors this matrix is combined with, the approximation is the least accurate for $\nu = (0.25, 0.25, 0.25, 0.25)$. If one queue receives a greater part of the total load, the approximation becomes better. Apparently the dependence between the queues is the strongest if the loads are uniformly distributed.

In contrast to the fact that the mean waiting time approximation with this p_{ij} and ν is the least accurate, it is striking that the approximation of the saturation loads is exact. This is easily seen if we observe that the sum of queue lengths is equal to the queue length in a $Geo^X/D/1$ queue with load λ and unit service times.

All queues are stable if and only if the $Geo^X/D/1$ queue is stable, which it is if $\lambda < 1$. Algorithm 3.3.1 gives precisely this value.

A second example for which the approximation is less accurate is the example with

$$(p_{ij}) = \begin{pmatrix} 0.28 & 0.29 & 0.15 & 0.28 \\ 0.34 & 0.03 & 0.15 & 0.48 \\ 0.15 & 0.21 & 0.30 & 0.34 \\ 0.40 & 0.04 & 0.01 & 0.55 \end{pmatrix},$$

and $\nu = (0.7, 0.1, 0.1, 0.1)$. We found that this example had the largest error in the saturation load approximation ($\sim 3\%$). While we cannot explain this error in particular, it seems very likely that a larger error in the saturation load approximation generally also induces a larger error in the mean waiting time approximation.

3.9 Conclusion

In this chapter we have devised approximations of throughput and stability conditions of small non-uniform switches. Furthermore, we approximated the mean waiting time in a switch by that in a $./Geo/1$ queue, with an arrival process equal to that of the switch; for a switch with Bernoulli arrivals, a $Geo/Geo/1$ queue is used, for a switch with Markovian Arrival Processes (MAPs), a $MAP/Geo/1$ queue is used, etc. The throughput and stability approximations only depend on the arrival rates, not on the specific arrival process. The mean waiting time approximation does depend on the specific arrival process.

We performed numerical experiments on a single example switch with Bernoulli arrivals, correlated arrivals (i.e., an arrival makes it more likely that there will be another arrival in the next time slot), and correlated destinations (i.e., consecutive packets are more likely to have the same destination). In addition to this, we performed numerical experiments on a much larger scale for switches with Bernoulli arrivals. These numerical investigations suggest that the approximations are very accurate in general.

Chapter 4

Reduction of polling tree networks

In the remaining chapters, we consider concentrating tree networks of polling stations. Although this model is primarily motivated by networks on chips where all traffic has the same destination (e.g., multiple masters sharing a single slave), the range of applications for which this model is suitable is much broader; for instance a number of workstations connected to a single server via a number of routers, a manufacturing environment where all jobs require the same final operation after a number of intermediate operations, etc.

We assume all packets in the network have unit size and arrive from the exterior according to independent batch Bernoulli arrival processes. All packets are eventually routed to the same node (called node 0), from which they leave the network. The service disciplines of all nodes are work-conserving and the service discipline of node 0 has to be *HoL-based* as well, which is an assumption that is satisfied by, a.o., m_i-limited service, exhaustive service, and priority disciplines.

Let a type i packet be a packet that eventually visits queue i of node 0. In this chapter, we establish a distributional relation between the number of type i packets in the network and in a *single* station system, and we show equality of the mean end-to-end delay of type i packets in the two systems. Essentially this reduces an arbitrary tree network to a much simpler system of one node, while preserving the mean end-to-end delay of type i packets.

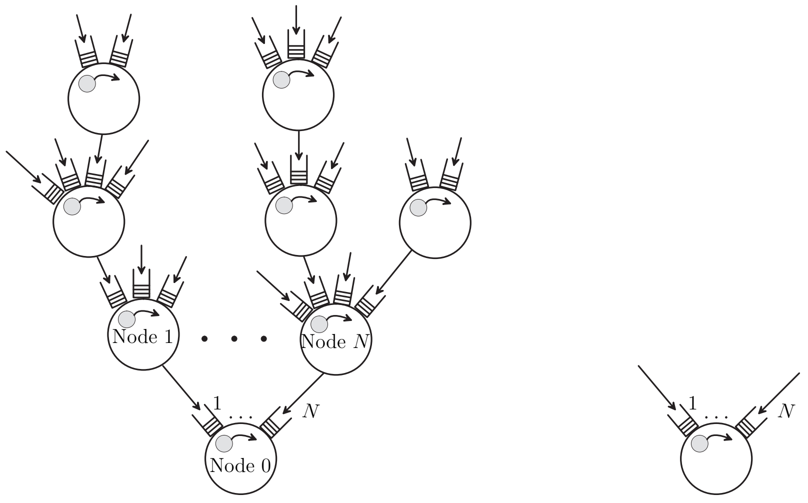
4.1 Introduction

We consider a concentrating tree network with an arbitrary number of nodes, as depicted in Figure 4.1a. Every node is a polling system with an arbitrary number of queues. Packets of fixed size 1 arrive from the exterior of the network to each node, where they are stored and eventually transmitted to the next node. All packets leave the network via the same node (the sink), which we call node 0.

Node 0 is a single server polling system with N queues. For the other nodes the number of queues is unspecified. Queue i of node 0 may store packets arriving from some node upstream or from the exterior directly. In the first case, we say that node i is the node connected to queue i of node 0. In the second case, node i does not exist. For the other nodes in the network it is unimportant which number identifies them. We refer to packets that pass through queue i of node 0 as 'type i' packets, for $i = 1, \ldots, N$.

All nodes have work-conserving service disciplines (i.e., the server has to serve precisely one packet if there is at least one and it may not create or remove additional packets). Except for node 0, the service disciplines are otherwise arbitrary.

The service discipline of node 0 is not only work-conserving, but also *HoL-based*. This means that the decision which queue is served may only depend on whether or not queues are occupied, and not on the *number* of packets present in each queue. Examples of HoL-based service disciplines are 1-limited service (serve one



(a) The original system **(b)** The reduced system

Figure 4.1: A schematic representation of the reduction. We consider an arbitrary tree network for which Assumption 4.2.1 is satisfied and reduce it to a single node while preserving the mean end-to-end delay of type i packets. In the figure all queues of node 0 store packets coming from nodes upstream, but it is also possible that they store packets directly arriving from the exterior (but not both).

packet per queue), exhaustive service (keep serving a queue until it is empty), and priority disciplines (serve a packet from the queue with the highest priority if it is non-empty, otherwise consider the queue with second highest priority, and so on). Service disciplines that are not HoL-based are disciplines such as gated service and longest or shortest queue first.

We construct a system consisting of one node (see Figure 4.1b), called the reduced system. The reduced system uses the same service discipline as node 0 and its arrival processes are given by superpositions of the arrivals to nodes upstream of node i. We establish a distributional relation between the contents of the network and the reduced system and we show that the mean end-to-end delay of type i packets in the network and the reduced system are equal.

The reason why the service discipline of node 0 has to be HoL-based for our results to be applicable is the following: As the arrival processes to the reduced system are given by the superposition of arrival processes in the network, the number of packets in queue i of node 0 in the original system is typically smaller than the number of packets in queue i of the reduced system. If the server is allowed to make decisions based on the number of packets in the queues, the behaviour in both systems might be entirely different, which would hence lead to different mean end-to-end delays.

Since HoL-based service disciplines may not use any information other than whether or not queues are empty, we can show that the behaviour in the reduced and original system is stochastically identical if a HoL-based service discipline is used. We do so by introducing a *coupled* system with one node, of which queue i is empty if and only if queue i of node 0 of the original system is empty. In Section 4.4 we give an example without a HoL-based service discipline for which the reduction fails.

There is a large amount of literature available on reductions of networks. The earliest results are those of Avi-Itzhak [9] and Friedman [57], where a tandem network of deterministic multi-server queues with an arbitrary arrival process is considered and reduced to a single node. Rubin [121, 122] studies a tandem network that is based on packet-switched communication networks. Mandjes and Van Uitert [97] apply the results of Avi-Itzhak and Friedman to obtain results for the overflow probability in a two-queue tandem network.

These papers concern tandem networks without cross-traffic (i.e., with a single source of arrivals). Rubin [123] devises an approximation for a tandem network with deterministic service times and cross traffic. Shalmon and Kaplan [127] perform an exact analysis of a tandem network with Poisson arrivals and deterministic packet sizes with FIFO service order. Shalmon [126] extends this exact analysis to a system with fixed priority or exhaustive service disciplines. Furthermore, Shalmon [126] establishes a reduction result, related to ours, for a system with general arrival processes and fixed priority or exhaustive service disciplines. Neely, Rohrs and Modiano [105] analyse a continuous-time concentrating tree network with general arrival processes at each of the sources and reduce this to a two-stage equivalent network.

Finally, Morrison [104] considers a tree network where all packets pass through at most two nodes before leaving the network. He constructs a reduced system consisting of one node and establishes a sample path relation between the queue contents in both systems. Morrison argues that this reduction can be applied repeatedly to arbitrary tree networks but restricts his analysis to arbitrary tandem queues.

At the heart of our reduction result is a deterministic relation on sample paths as well. This relation is an extension of Morrison's since the latter is a relation between the *total* contents of the reduced system and the network, while we establish a relation between the number of type i packets in the reduced system and the network. For HoL-based service disciplines, Morrison's result follows from our sample path relation by summing over all i. In addition to this, we address the precise implications of this relation for a network with stochastic arrivals.

This chapter is organised as follows: In Section 4.2 we describe the model and the reduced system in more detail and we state our main results. These results are then proved in Section 4.3. In Section 4.4 we give an example that shows why HoL-based service disciplines are required. A conclusion is drawn in Section 4.5.

4.2 Formalisation

In Section 4.2.1, we formalise the network model and the condition the service discipline of node 0 has to satisfy. In Section 4.2.2 we define the reduced system and present a more precise formulation of the main result.

4.2.1 The original system

We define $\mathcal{N}^{(m)}$ as the set of nodes whose output reaches node m at some point in time, combined with node m itself. Recall that if queue $i = 1, \ldots, N$ of node 0 stores packets directly arriving from the exterior, node i does not exist. In this case, we define $\mathcal{N}^{(i)} = \emptyset$.

We denote the total number of packets (i.e., summed over all queues) arriving from the exterior to node m at time t by $X_t^{(m)}$. For all queues i of node 0 that store packets directly arriving from the exterior, we denote the number of arriving packets by $X_{i,t}^{(0)}$. We assume that the external arrivals are governed by independent batch Bernoulli processes, i.e., $X_t^{(m)}$ and $X_{i,t}^{(0)}$ are both i.i.d. with respect to t, with generic random variables $X^{(m)}$ and $X_i^{(0)}$, and expectation $\lambda^{(m)}$ and $\lambda_i^{(0)}$. The notation used here is indicative of the notation used throughout the chapter: We add the node as a superscript and the queue (if applicable) as subscript.

The total arrival rate at node m, $\gamma^{(m)}$, comprises both external arrivals and arrivals from nodes upstream. Observe that every packet arriving at node m must have arrived from the exterior at some node upstream or at m itself:

$$\gamma^{(m)} = \sum_{l \in \mathcal{N}^{(m)}} \lambda^{(l)} \qquad \text{for all } m. \tag{4.2.1}$$

We assume $\gamma^{(0)} < 1$ so all queues are stable.

We denote the number of packets in all queues of node $m \neq 0$ at time t by $Q_t^{(m)}$. Arrivals take place at the end of time with departures before arrivals (according to the la-df model). A packet arriving at the end of time slot $[t-1, t)$ is not in the queue before time t. A packet that arrives at time t may be served in time slot $[t, t+1)$ and leave at time $t+1$. For convenience we assume that the network starts operating at time 0: If $t \leq 0$ then $X_t^{(m)} = 0$. Consequently, $Q_t^{(m)} = 0$ for all $t \leq 0$. We denote the steady state queue lengths by dropping the subscript t: $Q_i^{(0)}$ and $Q^{(m)}$.

We denote the queue the server of node 0 serves in $[t, t+1)$ (or the position the server moves to at time t) by $P_t^{(0)} \in \{0, \dots, N\}$, where $P_t^{(0)} = 0$ if the server is idle in $[t, t+1)$. Precisely one packet departs from queue i at time $t+1$ if and only if $P_t^{(0)} = i$, so

$$Q_{i,t+1}^{(0)} = \begin{cases} Q_{i,t}^{(0)} + \varepsilon(Q_t^{(i)}) - 1(P_t^{(0)} = i) & \text{if } \mathcal{N}^{(i)} \neq \emptyset \\ Q_{i,t}^{(0)} + X_{i,t+1}^{(0)} - 1(P_t^{(0)} = i) & \text{if } \mathcal{N}^{(i)} = \emptyset \end{cases} \tag{4.2.2}$$

where $Q_{i,t}^{(0)}$ is the length of queue i at time t, $1(\cdot)$ is the indicator function, $\varepsilon(x) = 1$ if $x > 0$, and $\varepsilon(0) = 0$.

We define a vector $\boldsymbol{Q}_t^{(0)} = (Q_{1,t}^{(0)}, \dots, Q_{N,t}^{(0)})$ containing all the queue lengths, and $\varepsilon(\boldsymbol{Q}_t^{(0)}) = (\varepsilon(Q_{1,t}^{(0)}), \dots, \varepsilon(Q_{N,t}^{(0)}))$.

Assumption 4.2.1. Node 0 uses a HoL-based service discipline, which means that it satisfies

$$\mathbb{P}\Big(P_t^{(0)} = j \Big| P_{t-1}^{(0)}, \dots, P_1^{(0)},\ \boldsymbol{Q}_t^{(0)}, \dots, \boldsymbol{Q}_1^{(0)}\Big) \\ = \mathbb{P}\Big(P_t^{(0)} = j \Big| P_{t-1}^{(0)}, \dots, P_{t-M}^{(0)},\ \varepsilon(\boldsymbol{Q}_t^{(0)}), \dots, \varepsilon(\boldsymbol{Q}_{t-M}^{(0)})\Big),$$

for some finite M. In other words, the server makes a (random) decision which queue it starts serving at time t based on whether there were packets in the queues and the queues it served during a history of M time slots.

Two main classes of service disciplines are HoL-based: The first is Bernoulli scheduling, where after a service completion at queue i, the server decides to serve queue i again with probability q_i and moves to one of the other queues with probability $1 - q_i$. This class also contains the 1-limited and exhaustive service disciplines as extreme cases ($q_i = 0$ and $q_i = 1$ respectively). The second main class is the class of m_i-limited service disciplines, where the server serves at most m_i packets from a queue before moving to one of the other queues. The history parameter M ensures that this is allowed; provided $m_i \leq M$, the number of packets served consecutively is contained in $P_{t-1}^{(0)}, \dots, P_{t-M}^{(0)}$.

If the server moves to one of the other queues it may choose among the non-empty queues according to some fixed order, such as round robin, a polling table, or a priority index, or according to a random order that is independent of the queue lengths.

Service disciplines that are not HoL-based for instance take arrival times or deadlines into account (e.g., earliest arrival time first, earliest deadline first), or queue lengths (e.g., gated service and shortest/longest queue first). In Section 4.4 we give an example that shows why our reduction fails if Assumption 4.2.1 is violated.

Remark 4.2.2. The history parameter M has to be finite due to a technicality. We will come back to this issue in Remark 4.3.6.

Remark 4.2.3. Assumption 4.2.1 also prevents the server of node 0 to take into account the size of the batch in which a packet arrived, since this information is not contained in $\varepsilon(\boldsymbol{Q}_{t-M}^{(0)}), \dots, \varepsilon(\boldsymbol{Q}_t^{(0)})$.

Remark 4.2.4. We have not specified in which order packets are served *within* each queue (for instance FIFO, LIFO, etc.). We only specify that a packet has to be served from a queue that is selected according to Assumption 4.2.1. The specific order in which packets are served within a queue is irrelevant for our results.

The mean sojourn time of packets in node m is denoted by $\mathbb{E}[S^{(m)}]$, for all m. We also define $\mathbb{E}[S_i^{(0)}]$ as the mean sojourn time of a packet in queue i of node 0. The corresponding mean waiting times (or delays) are denoted by a W: $\mathbb{E}[W^{(m)}] = \mathbb{E}[S^{(m)}] - 1$ and $\mathbb{E}[W_i^{(0)}] = \mathbb{E}[S_i^{(0)}] - 1$.

In order to describe the end-to-end delay of type i packets, we introduce the set $P(m,l)$ containing every node on the path from m to l, including m and l themselves. That is,

$$P(m,l) = \{k : k \in \mathcal{N}^{(l)} \text{ and } m \in \mathcal{N}^{(k)}\} \qquad \text{for all } l \text{ and } m \in \mathcal{N}^{(l)}. \tag{4.2.3}$$

The distance from node m to l, with $m \in \mathcal{N}^{(l)}$, is given by

$$d(m,l) = |P(m,l)| - 1,$$

so that $d(m,m) = 0$, $d(m,l) = 1$ if the output of m goes to l directly, $d(m,l) = 2$ if there is one intermediate node, and so on.

The mean end-to-end delay of type i packets is now given by

$$\mathbb{E}[Z_i] = \sum_{m \in \mathcal{N}^{(i)}} \frac{\lambda^{(m)}}{\gamma^{(i)}} \sum_{l \in P(m,i)} \mathbb{E}[W^{(l)}] + \mathbb{E}[W_i^{(0)}].$$

The reasoning behind this equation is the following: A fraction $\lambda^{(m)}/\gamma^{(i)}$ of the packets that arrive at queue i of node 0, arrived from the exterior at node m. The mean end-to-end delay of these packets is the sum of waiting times at nodes from m to i. Note that if $\mathcal{N}^{(i)} = \emptyset$, $\mathbb{E}[Z_i] = \mathbb{E}[W_i^{(0)}]$, i.e., if packets arrive from the exterior to queue i of node 0 directly, their mean end-to-end delay is given only by their mean waiting time in queue i of node 0.

4.2.2 The reduced system

In this section we describe the construction of the reduced system and state our main result. All quantities related to the reduced system are denoted by primes next to the normal letters. We will refer to the network formalised in Section 4.2.1 as the original system.

The arrivals to the reduced system are constructed by aggregating all arrivals to the node i subtree of the original system ($i = 1, \dots, N$) into a single stream. We let $X'_{i,t}$ denote the number of arrivals to queue i of the reduced system at time t. The random variables $X'_{i,t}$ are i.i.d. with respect to t, with generic random variable X'_i given by

$$X'_i \overset{d}{=} \begin{cases} \sum\limits_{m \in \mathcal{N}^{(i)}} X^{(m)}, & \text{if } \mathcal{N}^{(i)} \neq \emptyset, \\ X_i^{(0)}, & \text{if } \mathcal{N}^{(i)} = \emptyset. \end{cases}$$

The evolution of the queues can now be described as follows:

$$Q'_{i,t+1} = Q'_{i,t} + X'_{i,t+1} - 1(P'_t = i),$$

where P'_t is the position of the server of the reduced system at time t, and $Q'_{i,t}$ is the length of queue i at time t. We denote the vector of all queue lengths at time t by $\boldsymbol{Q}'_t = (Q'_{1,t}, \dots, Q'_{N,t})$. The steady state length of queue i is denoted by Q'_i.

The service discipline of the reduced system is the same HoL-based discipline as that of node 0 in the original system. In practice, this means that if node 0 uses m_i-limited, then so must the reduced system, if node 0 uses exhaustive service, then so must the reduced system, etc. In a more general setting, we mean the following: Given identical server positions and identical empty and non-empty queues during the last M time slots, the probability that a certain queue is served is the same under both service disciplines:

$$\mathbb{P}\Big(P'_t = p_t |\ \varepsilon(\boldsymbol{Q}'_s) = \varepsilon(\boldsymbol{q}_s),\ \text{for all } t - M \le s \le t,\ \ P'_s = p_s,\ \text{for all } t - M \le s < t\Big) =$$
$$\mathbb{P}\Big(P_t^{(0)} = p_t |\ \varepsilon(\boldsymbol{Q}_s^{(0)}) = \varepsilon(\boldsymbol{q}_s),\ \text{for all } t - M \le s \le t,\ \ P_s^{(0)} = p_s,\ \text{for all } t - M \le s < t\Big). \tag{4.2.4}$$

The mean waiting time of a type i packet in the reduced system is denoted by $\mathbb{E}[W'_i]$. Our two main results are the following:

Theorem 4.2.5. *In steady-state, the length of queue i of the reduced system is in distribution equal to the number of type i packets in the network, minus external arrivals to node m in the last $d(m, 0)$ time slots, summed over all m:*

$$Q'_i \overset{d}{=} \lim_{t \to \infty} \left(\sum_{m \in \mathcal{N}^{(i)}} Q_t^{(m)} + Q_{i,t}^{(0)} - \sum_{m \in \mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} X_{t+1-d}^{(m)} \right). \tag{4.2.5}$$

Theorem 4.2.6 (Reduction theorem)**.** *The mean end-to-end delay (total waiting time) of type i packets is the same in the original and the reduced system:*

$$\mathbb{E}[Z_i] = \mathbb{E}[W'_i].$$

Remark 4.2.7. Theorem 4.2.5 also implies that the number of type i packets in the single-node system is stochastically smaller than that in the network: $Q_i' \leq_d \sum_{m\in\mathcal{N}^{(i)}} Q^{(m)} + Q_i^{(0)}$. We expect that this bound is tight in heavy traffic.

4.3 Proof of the main results

In this section, we prove Theorem 4.2.5 and 4.2.6. In order to do so, we use a coupling argument: In Section 4.3.1, we introduce another single station system, called the *coupled* system. The arrivals to this system are constructed in a deterministic way from the arrivals to the original system. This deterministic construction allows us to prove a sample path relation between the *original* system and the *coupled* system in Section 4.3.2. In Section 4.3.3 we show that the queue lengths of the *coupled* system and the *reduced* system are equal in distribution. These two relations combined provide us with a relation between the *original* and *reduced* system, namely Theorem 4.2.5. In Section 4.3.4 we prove Theorem 4.2.6 by applying Little's law to Theorem 4.2.5.

4.3.1 The coupled system

In this section, we describe the coupled system. All quantities related to this system are denoted by tildes above the normal letters. We define

$$\widetilde{X}_{i,t} = \begin{cases} \sum\limits_{m\in\mathcal{N}^{(i)}} X^{(m)}_{t-d(m,0)}, & \text{if } \mathcal{N}^{(i)} \neq \emptyset, \\ X^{(0)}_{i,t}, & \text{if } \mathcal{N}^{(i)} = \emptyset. \end{cases} \tag{4.3.1}$$

The evolution of the queues can now be described as follows:

$$\widetilde{Q}_{i,t+1} = \widetilde{Q}_{i,t} + \widetilde{X}_{i,t+1} - 1(\widetilde{P}_t = i), \tag{4.3.2}$$

where $\widetilde{P}_t$ is the position of the server. We denote the vector of all queue lengths at time t by $\widetilde{\boldsymbol{Q}}_t = (Q_{1,t}, \dots, Q_{N,t})$.

We couple the service discipline to that of node 0 in the original system in the following way: If both servers have the same history, their next position will also be the same, i.e., if $\varepsilon(\widetilde{\boldsymbol{Q}}_s) = \varepsilon(\boldsymbol{Q}_s^{(0)})$, for all $t-M \leq s \leq t$, and $\widetilde{P}_s = P_s^{(0)}$, for all $t-M \leq s < t$, then $\widetilde{P}_t = P_t^{(0)}$. In Section 4.3.2 we prove that $\widetilde{P}_t = P_t^{(0)}$ and $\varepsilon(\widetilde{\boldsymbol{Q}}_t) = \varepsilon(\boldsymbol{Q}_t^{(0)})$ for all t.

The intuition behind this coupling is that a packet reaches node 0 in the original system after or at the same time it arrives to the coupled system; a packet requires *at least* $d(m, 0)$ time slots to reach node 0 from node m, which is precisely the number of time slots by which an arrival to the coupled system is delayed. Using induction we can show that queue i of the coupled system and node 0 of the original system are empty at precisely the same times. Together with the (inductive) definition of $\widetilde{P}_t$ this implies a sample path relation between the *original* system and the *coupled* system for all t.

Furthermore, the steady state queue lengths of the *coupled* and *reduced* systems are stochastically the same due to the following argument: By the nature of batch Bernoulli arrival processes, imposing a time-delay does not stochastically change them. In addition to this, the service disciplines of the coupled and the reduced system are shown to be stochastically equal. As the arrival processes and service disciplines of the coupled and reduced system are stochastically identical, their queue lengths must be so too.

4.3.2 The original and the coupled system

We establish a sample path relation between the original and the coupled system for all t, which leads to a similar steady state relation. We first give Lemma 4.3.1 for further reference. Next, we obtain our sample path relation in Proposition 4.3.2. Taking limits then yields the steady state relation (Corollary 4.3.3).

We first describe the evolution of the *total* contents of the nodes. In order to do so, we define $\mathcal{N}_1^{(m)}$ as the set of nodes whose output enters node m directly, i.e.,

$$\mathcal{N}_1^{(m)} = \{l \in \mathcal{N}^{(m)} : d(l,m) = 1\}, \qquad \text{for all } m.$$

The total number of packets arriving to node m is given by the number of external arrivals plus the arrivals from nodes upstream of m, and one packet is served if there is at least one packet present, so for all m,

$$Q_{t+1}^{(m)} = Q_t^{(m)} + X_{t+1}^{(m)} + \sum_{l \in \mathcal{N}_1^{(m)}} \varepsilon(Q_t^{(l)}) - \varepsilon(Q_t^{(m)}). \tag{4.3.3}$$

The following lemma is presented for further reference:

Lemma 4.3.1. *For all t and $i = 1, \ldots, N$,*

$$\sum_{m \in \mathcal{N}^{(i)}} Q_t^{(m)} = \sum_{m \in \mathcal{N}^{(i)}} Q_{t-d(m,0)}^{(m)} + \sum_{m \in \mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} X_{t+1-d}^{(m)} - \sum_{m \in \mathcal{N}^{(i)}} \varepsilon(Q_{t-d(m,0)}^{(m)}). \tag{4.3.4}$$

Proof. We first observe that, by (4.3.3),

$$Q_t^{(m)} - Q_{t-1}^{(m)} = X_t^{(m)} - \varepsilon(Q_{t-1}^{(m)}) + \sum_{l \in \mathcal{N}_1^{(m)}} \varepsilon(Q_{t-1}^{(l)}).$$

If we apply this argument $d(m,0)$ times, we obtain

$$Q_t^{(m)} - Q_{t-d(m,0)}^{(m)} = \sum_{d=1}^{d(m,0)} \left(X_{t+1-d}^{(m)} - \varepsilon(Q_{t-d}^{(m)}) + \sum_{l \in \mathcal{N}_1^{(m)}} \varepsilon(Q_{t-d}^{(l)}) \right),$$

for all m. Summing over all $m \in \mathcal{N}^{(i)}$ yields

$$\sum_{m\in\mathcal{N}^{(i)}} Q_t^{(m)} = \sum_{m\in\mathcal{N}^{(i)}} Q_{t-d(m,0)}^{(m)} + \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} X_{t+1-d}^{(m)} - \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} \varepsilon(Q_{t-d}^{(m)}) + \sum_{m\in\mathcal{N}^{(i)}} \sum_{l\in\mathcal{N}_1^{(m)}} \sum_{d=1}^{d(m,0)} \varepsilon(Q_{t-d}^{(l)}). \tag{4.3.5}$$

Observe that in the double summation over the nodes in the last term of (4.3.5), we include for each $m \in \mathcal{N}^{(i)}$ all l immediately upstream of m. We thus sum over all $l \in \mathcal{N}^{(i)}$, except $l = i$. As $d(m,0) = d(l,0) - 1$ for such m and l, we get

$$\sum_{m\in\mathcal{N}^{(i)}} \sum_{l\in\mathcal{N}_1^{(m)}} \sum_{d=1}^{d(m,0)} \varepsilon(Q_{t-d}^{(l)}) = \sum_{\substack{l\in\mathcal{N}^{(i)} \\ l\neq i}} \sum_{d=1}^{d(l,0)-1} \varepsilon(Q_{t-d}^{(l)}) = \sum_{l\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(l,0)-1} \varepsilon(Q_{t-d}^{(l)}), \tag{4.3.6}$$

since $d(i,0) = 1$. Substitution of the latter expression in (4.3.5) indeed yields (4.3.4). □

Proposition 4.3.2. *For all t and $i = 1, \dots, N$,*

$$\widetilde{Q}_{i,t} = \sum_{m\in\mathcal{N}^{(i)}} Q_t^{(m)} + Q_{i,t}^{(0)} - \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} X_{t+1-d}^{(m)}, \tag{4.3.7}$$

$$\varepsilon(\widetilde{Q}_{i,t}) = \varepsilon(Q_{i,t}^{(0)}), \tag{4.3.8}$$

$$\widetilde{P}_t = P_t^{(0)}. \tag{4.3.9}$$

Proof. We prove this lemma by induction. Note that (4.3.7)-(4.3.9) trivially hold for $t \leq 0$ due to the assumption that the systems are initially empty. We hypothesise that the proposition is true for $\dots, t-1, t$ and prove it for $t+1$. First, assume $\mathcal{N}^{(i)} \neq \emptyset$. By (4.3.1), (4.3.2), and the induction hypothesis,

$$\begin{aligned}
\widetilde{Q}_{i,t+1} &= \widetilde{Q}_{i,t} + \widetilde{X}_{i,t+1} - 1(\widetilde{P}_t = i) \\
&= \sum_{m\in\mathcal{N}^{(i)}} Q_t^{(m)} + Q_{i,t}^{(0)} - \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} X_{t+1-d}^{(m)} + \sum_{m\in\mathcal{N}^{(i)}} X_{t+1-d(m,0)}^{(m)} - 1(P_t^{(0)} = i) \\
&= \sum_{m\in\mathcal{N}^{(i)}} Q_t^{(m)} + Q_{i,t}^{(0)} - \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)-1} X_{t+1-d}^{(m)} - 1(P_t^{(0)} = i).
\end{aligned}$$

By applying (4.2.2) and (4.3.3) we obtain

$$\begin{aligned}
\widetilde{Q}_{i,t+1} &= \sum_{m\in\mathcal{N}^{(i)}} \Big(Q_{t+1}^{(m)} + \varepsilon(Q_t^{(m)}) - X_{t+1}^{(m)} - \sum_{l\in\mathcal{N}_1^{(m)}} \varepsilon(Q_t^{(l)})\Big) + Q_{i,t+1}^{(0)} - \varepsilon(Q_t^{(i)}) \\
&\quad - \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)-1} X_{t+1-d}^{(m)} \\
&= \sum_{m\in\mathcal{N}^{(i)}} Q_{t+1}^{(m)} + \sum_{m\in\mathcal{N}^{(i)}} \varepsilon(Q_t^{(m)}) - \sum_{m\in\mathcal{N}^{(i)}} \sum_{l\in\mathcal{N}_1^{(m)}} \varepsilon(Q_t^{(l)}) + Q_{i,t+1}^{(0)} - \varepsilon(Q_t^{(i)}) \\
&\quad - \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=0}^{d(m,0)-1} X_{t+1-d}^{(m)}.
\end{aligned}$$

We proceed by interchanging the order of summation in the first double sum. Similar to the proof of Lemma 4.3.1, we include for all $m \in \mathcal{N}^{(i)}$ all l immediately upstream of m. This implies that we sum over all $l \in \mathcal{N}^{(i)}$, except $l = i$. We thus get (cf. (4.3.6))

$$\sum_{m\in\mathcal{N}^{(i)}} \sum_{l\in\mathcal{N}_1^{(m)}} \varepsilon(Q_t^{(l)}) = \sum_{\substack{l\in\mathcal{N}^{(i)} \\ l\neq i}} \varepsilon(Q_t^{(l)}) = \sum_{l\in\mathcal{N}^{(i)}} \varepsilon(Q_t^{(l)}) - \varepsilon(Q_t^{(i)}),$$

which yields (4.3.7).

Now assume $\mathcal{N}^{(i)} = \emptyset$. Then,

$$\begin{aligned}
\widetilde{Q}_{i,t+1} &= \widetilde{Q}_{i,t} + \widetilde{X}_{i,t+1} - 1(\widetilde{P}_t = i), && \text{by (4.3.2)}, \\
&= \widetilde{Q}_{i,t} + X_{i,t+1}^{(0)} - 1(\widetilde{P}_t = i), && \text{by (4.3.1)}, \\
&= Q_{i,t}^{(0)} + X_{i,t+1}^{(0)} - 1(P_t^{(0)} = i), && \text{by the induction hypothesis}, \\
&= Q_{i,t+1}^{(0)}, && \text{by (4.2.2)}.
\end{aligned}$$

To prove (4.3.8), we first prove that $\widetilde{Q}_{i,t+1} = 0$ implies $Q_{i,t+1}^{(0)} = 0$ and then that $Q_{i,t+1}^{(0)} = 0$ implies $\widetilde{Q}_{i,t+1} = 0$. Note that if $\mathcal{N}^{(i)} = \emptyset$, (4.3.8) immediately follows from (4.3.7), so we restrict our proof to i for which $\mathcal{N}^{(i)} \neq \emptyset$. From (4.3.4) and (4.3.7) with $t+1$ substituted for t, it follows that

$$\widetilde{Q}_{i,t+1} = Q_{i,t+1}^{(0)} + \sum_{m\in\mathcal{N}^{(i)}} \Big(Q_{t+1-d(m,0)}^{(m)} - \varepsilon(Q_{t+1-d(m,0)}^{(m)})\Big). \tag{4.3.10}$$

Hence $\widetilde{Q}_{i,t+1} = 0$ implies that $Q_{t+1-d(m,i)}^{(m)} = \varepsilon(Q_{t+1-d(m,i)}^{(m)})$ for all $m \in \mathcal{N}^{(i)}$ and $Q_{i,t+1}^{(0)} = 0$.

Suppose now that $Q_{i,t+1}^{(0)} = 0$. This means that $0 = Q_{i,t+1}^{(0)} = Q_{i,t}^{(0)} + \varepsilon(Q_t^{(i)}) - 1(P_t^{(0)} = i)$, which entails that $Q_t^{(i)} = 0$. We can then apply this argument to $Q_t^{(i)} =$

$Q_{t-1}^{(i)} - \varepsilon(Q_{t-1}^{(i)}) + X_t^{(i)} + \sum_{m\in\mathcal{N}_1^{(i)}} \varepsilon(Q_{t-1}^{(m)})$ to obtain $Q_{t-1}^{(m)} = 0$ for $m \in \mathcal{N}_1^{(i)}$, and so on. This eventually results in $Q_{t+1-d(m,0)}^{(m)} = 0$ for all m. Together with (4.3.10) we conclude $\widetilde{Q}_{i,t+1} = 0$.

In Section 4.3.1 we defined $\widetilde{P}_t$ such that if $\widetilde{P}_s = P_s^{(0)}$ for all $s = t-M, \ldots, t-1$, and $\varepsilon(\widetilde{\boldsymbol{Q}}_s) = \varepsilon(\boldsymbol{Q}_s^{(0)})$ for all $s = t-M, \ldots, t$, then $\widetilde{P}_t = P_t^{(0)}$. This definition implies that $\widetilde{P}_{t+1} = P_{t+1}^{(0)}$, which completes the proof. □

Corollary 4.3.3. *The following relation holds:*

$$\widetilde{Q}_i \overset{d}{=} \lim_{t\to\infty} \left(\sum_{m\in\mathcal{N}^{(i)}} Q_t^{(m)} + Q_{i,t}^{(0)} - \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} X_{t+1-d}^{(m)} \right). \tag{4.3.11}$$

Remark 4.3.4. Proposition 4.3.2 is actually much stronger than Corollary 4.3.3; Proposition 4.3.2 holds for all t, for any realisation of $X_t^{(m)}$, and hence regardless of the underlying arrival process. Proposition 4.3.2 thus gives a deterministic relation on sample paths. We use it here primarily to obtain Corollary 4.3.3, but in itself it is an extension of the reduction result of Morrison [104].

4.3.3 The reduced and the coupled system

In this section, we show that the steady state queue lengths of the coupled and the reduced system are the same (Proposition 4.3.5). By combining this with Corollary 4.3.3, we prove Theorem 4.2.5 at the end of the subsection.

Proposition 4.3.5. *The steady state queue lengths of the coupled and the reduced system are the same in distribution:*

$$Q_i' \overset{d}{=} \widetilde{Q}_i. \tag{4.3.12}$$

Proof. We introduce the discrete time process

$$Y_t' = (\boldsymbol{Q}_t', \varepsilon(\boldsymbol{Q}_{t-1}'), \ldots, \varepsilon(\boldsymbol{Q}_{t-M}'), P_{t-1}', \ldots, P_{t-M}'),$$

and $\widetilde{Y}_t$, defined similarly.

The processes Y_t' and $\widetilde{Y}_t$ are aperiodic, irreducible, and positive recurrent Markov chains, and therefore have a unique stationary distribution. This implies that $\widetilde{Q}_{i,t}$, and $Q_{i,t}'$, being components of these Markov chains, have unique stationary distributions too. To prove $Q_i' \overset{d}{=} \widetilde{Q}_i$ we show that the transition probabilities of Y_t' and $\widetilde{Y}_t$ are the same for $t > \max_m d(m,0)$. After all, if these Markov chains have the same transition probabilities, they have the same equilibrium distribution and hence so do $Q_{i,t}'$ and $\widetilde{Q}_{i,t}$.

Due to the time-delay of arrivals to the coupled system, and the assumption that $X_t^{(m)} = 0$ for $t \leq 0$, it can easily be checked that the transition probabilities of $\widetilde{Y}_t$ depend on t if $t \leq \max_m d(m,0)$. Nevertheless, this does not affect the equilibrium distribution of $\widetilde{Y}_t$ since it only affects a finite initial period of time. For $t > \max_m d(m,0)$ the transition probabilities of $\widetilde{Y}_t$ are time-homogeneous.

Let $\omega_t = \{\boldsymbol{q}_t, \varepsilon(\boldsymbol{q}_{t-1}), \ldots, \varepsilon(\boldsymbol{q}_{t-M}), p_{t-1}, \ldots, p_{t-M}\}$. Assume furthermore $t > \max_m d(m,0)$. We will show

$$\mathbb{P}(Y'_{t+1} = \omega_{t+1} | Y'_t = \omega_t) = \mathbb{P}(\widetilde{Y}_{t+1} = \omega_{t+1} | \widetilde{Y}_t = \omega_t).$$

Let $q_{j,t+1} = q_{j,t} - 1(p_t = j) + x_{j,t+1}$ for all j and t. Then, since the transition probabilities are only determined by the arrival probabilities and the HoL-based service discipline,

$$\begin{aligned}\mathbb{P}(Y'_{t+1} = \omega_{t+1} | Y'_t = \omega_t) \\ = \mathbb{P}\Big(P'_t = p_t \Big| \; \varepsilon(\boldsymbol{Q}'_s) = \varepsilon(\boldsymbol{q}_s), \;\; P'_s = p_s, \;\; \varepsilon(\boldsymbol{Q}'_t) = \varepsilon(\boldsymbol{q}_t)\Big) \\ \prod_{j=1}^{N} \mathbb{P}(X'_{j,t+1} = x_{j,t+1}),\end{aligned}$$

where $s = t-1, \ldots, t-M$. Using $X'_{j,t+1} \overset{d}{=} \widetilde{X}_{j,t+1}$ (see Section 4.3.1) and the fact that the service discipline is equal to that of node 0 (see (4.2.4)), we obtain

$$\begin{aligned}\mathbb{P}(Y'_{t+1} = \omega_{t+1} | Y'_t = \omega_t) \\ = \mathbb{P}\Big(P_t^{(0)} = p_t \Big| \; \varepsilon(\boldsymbol{Q}_s^{(0)}) = \varepsilon(\boldsymbol{q}_s), \;\; P_s^{(0)} = p_s, \;\; \varepsilon(\boldsymbol{Q}_t^{(0)}) = \varepsilon(\boldsymbol{q}_t)\Big) \\ \prod_{j=1}^{N} \mathbb{P}(\widetilde{X}_{j,t+1} = x_{j,t+1}).\end{aligned}$$

Finally, by $\varepsilon(\widetilde{\boldsymbol{Q}}_t) = \varepsilon(\boldsymbol{Q}_t^{(0)})$ and $\widetilde{P}_t = P_t^{(0)}$ for all t, we have

$$\begin{aligned}&\mathbb{P}(Y'_{t+1} = \omega_{t+1} | Y'_t = \omega_t) \\ &\quad = \mathbb{P}\Big(\widetilde{P}_t = p_t \Big| \; \varepsilon(\widetilde{\boldsymbol{Q}}_s) = \varepsilon(\boldsymbol{q}_s), \;\; \widetilde{P}_s = p_s, \;\; \varepsilon(\widetilde{\boldsymbol{Q}}_t) = \varepsilon(\boldsymbol{q}_t)\Big) \prod_{j=1}^{N} \mathbb{P}(\widetilde{X}_{j,t+1} = x_{j,t+1}) \\ &\quad = \mathbb{P}(\widetilde{Y}_{t+1} = \omega_{t+1} | \widetilde{Y}_t = \omega_t).\end{aligned}$$ □

Remark 4.3.6. In Section 4.2.1 we stated that $M < \infty$. This is necessary for the processes Y'_t and $\widetilde{Y}_t$ to have unique stationary distributions. If $M = \infty$ the queue that is served at time t might depend on the queue that was served at time 1 for all t. The transition probabilities of $\widetilde{Y}_t$ might thus depend on $\widetilde{Y}_1$ and the equilibrium distribution does not have to be unique.

The proof of Theorem 4.2.5 is now elementary:

Proof of Theorem 4.2.5. Theorem 4.2.5 follows immediately from Corollary 4.3.3 and Proposition 4.3.5. □

4.3.4 Waiting times

In this section we establish equality of the mean waiting times in the original and reduced system. The mean end-to-end sojourn time of type i packets passing through queue i of node 0 is given by

$$\mathbb{E}[T_i] = \sum_{m\in\mathcal{N}^{(i)}} \frac{\lambda^{(m)}}{\gamma^{(i)}} \sum_{l\in P(m,i)} \mathbb{E}[S^{(l)}] + \mathbb{E}[S_i^{(0)}].$$

The mean distance a type i packet must traverse before it reaches queue i of node 0 is given by

$$\mathbb{E}[D_i] = \sum_{m\in\mathcal{N}^{(i)}} \frac{\lambda^{(m)}}{\gamma^{(i)}} d(m,0).$$

Note that $\mathbb{E}[D_i]+1$ gives the mean total service time of type i packets.

We can now prove Theorem 4.2.6:

Proof of Theorem 4.2.6. First assume $\mathcal{N}^{(i)} \neq \emptyset$. In this case the total arrival rate to queue i of node 0 is given by $\gamma^{(i)}$. Using (4.2.5) and Little's law yields

$$\begin{aligned}
\mathbb{E}[S_i'] = \frac{1}{\gamma^{(i)}}\,\mathbb{E}[Q_i'] &= \sum_{m\in\mathcal{N}^{(i)}} \frac{1}{\gamma^{(i)}}\,\mathbb{E}[Q^{(m)}] + \frac{1}{\gamma^{(i)}}\,\mathbb{E}[Q_i^{(0)}] - \frac{1}{\gamma^{(i)}} \sum_{m\in\mathcal{N}^{(i)}} \sum_{d=1}^{d(m,0)} \mathbb{E}[X_d^{(m)}] \\
&= \frac{1}{\gamma^{(i)}} \sum_{m\in\mathcal{N}^{(i)}} \gamma^{(m)}\,\mathbb{E}[S^{(m)}] + \mathbb{E}[S_i^{(0)}] - \frac{1}{\gamma^{(i)}} \sum_{m\in\mathcal{N}^{(i)}} d(m,0)\lambda^{(m)} \\
&= \frac{1}{\gamma^{(i)}} \sum_{m\in\mathcal{N}^{(i)}} \sum_{l\in\mathcal{N}^{(m)}} \lambda^{(l)}\,\mathbb{E}[S^{(m)}] + \mathbb{E}[S_i^{(0)}] - \mathbb{E}[D_i],
\end{aligned}$$

where the last equation is a consequence of the fact that any arrival to node m must be an external arrival to some node $l \in \mathcal{N}^{(m)}$ (see (4.2.1)).

We continue by interchanging the order of summation. First, observe that for every $m \in \mathcal{N}^{(i)}$ we include all nodes l upstream of m, so that every $l \in \mathcal{N}^{(i)}$ is eventually included in the summation at least once. Second, for every $l \in \mathcal{N}^{(i)}$, we include all nodes m downstream of l and upstream of i, i.e., all m for which $m \in \mathcal{N}^{(i)}$ and $l \in \mathcal{N}^{(m)}$. This, however, corresponds precisely to the definition of the path from l to i (see (4.2.3)), so that

$$\mathbb{E}[S_i'] = \sum_{l\in\mathcal{N}^{(i)}} \frac{\lambda^{(l)}}{\gamma^{(i)}} \sum_{m\in P(l,i)} \mathbb{E}[S^{(m)}] + \mathbb{E}[S_i^{(0)}] - \mathbb{E}[D_i] = \mathbb{E}[T_i] - \mathbb{E}[D_i].$$

The equality of $\mathbb{E}[W_i']$ and $\mathbb{E}[Z_i]$, the mean end-to-end delay of type i packets, follows from the observation that $\mathbb{E}[D_i]+1$ is the mean total service time so $\mathbb{E}[Z_i]+\mathbb{E}[D_i]+1 = \mathbb{E}[T_i]$, combined with $\mathbb{E}[S_i'] = \mathbb{E}[W_i']+1$.

For $\mathcal{N}^{(i)} = \emptyset$ the proof is similar, except that now the arrival rate to queue i of node 0 is given by $\lambda_i^{(0)}$. Taking expectations of (4.2.5), dividing by $\lambda_i^{(0)}$, and applying Little's law immediately yields $\mathbb{E}[S_i'] = \mathbb{E}[T_i]$, which implies $\mathbb{E}[W_i'] = \mathbb{E}[Z_i]$. □

4.4 Discussion

In this section, we illustrate why a HoL-based service discipline is needed by means of an example. We consider a tree network consisting of two nodes (node 0 and 1), as depicted in Figure 4.2. Node 0 consists of two queues and node 1 of one. We assume that node 0 serves packets according to the shortest queue first policy (so the service discipline is *not* HoL-based). In case of equal queue lengths, it serves queue 1. The service policy of node 1 is arbitrary. We will show that the mean end-to-end delays of type 1 packets are different in the original and the reduced system.

We assume that batches of size $K > 1$ arrive to node 1, whereas batches of size 1 arrive to queue 2 of node 0:

$$X^{(1)} = \begin{cases} 0, & \text{w.p. } 1 - p_1, \\ K, & \text{w.p. } p_1, \end{cases}$$

$$X_2^{(0)} = \begin{cases} 0, & \text{w.p. } 1 - p_2, \\ 1, & \text{w.p. } p_2. \end{cases}$$

In the *original* system type 1 batches arrive to node 1 and are sent to node 0 packet-by-packet. Because node 0 serves queue 1 in case of equal queue lengths, type 1 packets never have to wait at node 0. The mean end-to-end delay of a type 1 packet is thus given by the mean waiting time in node 1, which is an ordinary $Geo^{X^{(1)}}/D/1$ queue.

In the *reduced* system type 1 batches arrive to the queues in their entirety. The length of queue 1 is therefore typically larger than that of queue 2 and queue 2 is usually served first (except if both queues have one packet). The mean waiting time of type 1 packets is thus equal to that in a $Geo^{X^{(1)}}/D/1$ queue plus some additional delay due to services of type 2 packets.

This example illustrates why a HoL-based service discipline is required; the number of packets in queue 1 of the reduced system is typically different from that of queue 1 of node 0, because batches arrive to the reduced system in their entirety, whereas they arrive to node 0 packet-by-packet. Even though the service discipline is shortest queue first in both systems, the server of the original system essentially

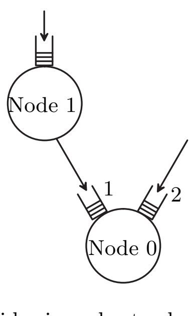


Figure 4.2: The network we consider in order to show that Assumption 4.2.1 is needed.

gives priority to type 1 and the server of the reduced system to type 2.

4.5 Conclusion

We have proved a distributional relation between queue lengths in a tree network and a reduced system consisting of one node, and equality of the mean end-to-end delay in both systems. Our results extend earlier work of Morrison [104], in the sense that our results hold conditioned on the type of the packet. By this we mean that where we prove a relation between the number of type i packets in both systems, [104] relates the *total* number of packets in both systems, i.e., summed over all i. This extension comes at the price that the class of allowed service disciplines is slightly more restrictive; the service discipline of node 0 has to be HoL-based.

A HoL-based service discipline is required because the number of packets in queue i of the reduced system is typically different from that in queue i of node 0. If the server is allowed to choose a queue based on the number of packets in a queue rather than whether or not it is empty, the behaviour of the systems might be entirely different, which could in turn lead to different mean waiting times.

The reduction theorem facilitates the analysis of a polling network through a simpler analysis of a single node. Even if the mean waiting time per queue in the single station polling system is unknown (such as for m_i-limited systems) the reduction result entails that single-station approximations can also be applied to networks without additional loss of accuracy in order to compute the mean end-to-end delay of type i packets.

Chapter 5

End-to-end delays in polling tree networks

In this chapter, we analyse mean end-to-end delays in concentrating tree networks of polling stations. The model we consider is the same as that of Chapter 4, except that *all* nodes use a HoL-based service discipline rather than only node 0.

We obtain an exact expression for the mean end-to-end delay averaged over all sources and an approximation of the mean end-to-end delay per source. The essential steps in this approximation are (i) the assumption that all streams passing through a certain queue at a node have the same mean waiting time in that node, and (ii) the reduction theorem of Chapter 4 (Thm. 4.2.6).

In the approximation, we express the mean end-to-end delay per source in terms of the mean waiting time (per queue) in single-station polling systems. Depending on the service disciplines used, the mean waiting time in these single-station polling systems can either be determined exactly or has to be approximated. In Chapter 6, we derive a new approximation that can also be used to obtain single-station results. This approximation is derived for a large subclass of HoL-based service disciplines, namely Bernoulli service combined with Markovian routing.

Although our approximation is derived for the entire class of HoL-based service disciplines, we are especially interested in polling stations with 1-limited service discipline used in switches in networks on chips. We furthermore show how our approximation can be applied to a model of a network on chip with four switches. Finally, we show that, for trees satisfying certain symmetry conditions, the mean end-to-end delay per type can be obtained exactly.

5.1 Model

We consider a concentrating tree network operating in discrete time as displayed in Figure 5.1. All packets have size 1 and arrive from external sources in batches according to independent batch Bernoulli arrival processes. The network operates under the late arrival - departures first model (see Section 1.3.2). A packet arriving at a node at the end of time slot $[t-1, t)$, i.e., at time t, may be served in time slot $[t, t+1)$. In this case it reaches the next node at time $t+1$ where it may be served in time slot $[t+1, t+2)$, and so on.

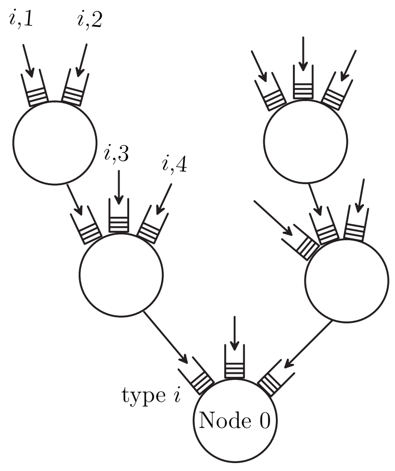


Figure 5.1: A polling tree network.

All nodes in the network are polling nodes without switch-over times. Node 0 is a node with N queues and is the last node of the network (the sink). All packets in the network must eventually pass through it and leave the network after that. Every node n in the tree network is itself the last node in a smaller tree network consisting of all nodes above node n and node n itself. We call the latter network the node n *subtree*.

We call a packet that eventually passes through queue i of node 0 a 'type i' packet. There are N_i external sources from which type i packets arrive. We subdivide type i packets into 'type i,j' packets, $j = 1, \dots, N_i$, such that the type denotes the source from which packets arrive (see Fig. 5.1). The set of type i packets is thus the union of the sets of type i,j packets.

The size of the batches of type i,j packets arriving each time slot is given by an arbitrary discrete non-negative random variable, denoted by $X_{i,j}$. We further define $X_i = \sum_{j=1}^{N_i} X_{i,j}$, and $X = \sum_i X_i$. We denote the expected batch sizes by $\rho_{i,j}$, ρ_i, and ρ, respectively. We assume $\rho < 1$, which implies that all nodes are stable because all packets have size 1.

The reduction theorem (Theorem 4.2.6) states that, if node 0 uses a HoL-based service discipline, the tree network can be reduced to a single-station polling system,

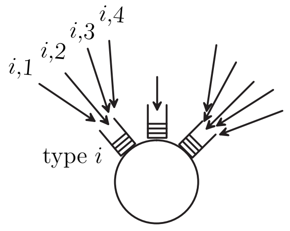


Figure 5.2: The reduced system.

called the *reduced* system (see Fig. 5.2). The reduced system is a system with arrival processes that are given by superpositions of the original arrival processes, i.e., it is a system with arrivals $X_i = \sum_j X_{i,j}$ to queue i, $i = 1, \dots, N$. The service discipline of the reduced system is the same as that of node 0 in the original system. If we denote the end-to-end delay of type i packets by Z_i, and the waiting time in queue i of the reduced system by W'_i, we thus have:

$$\mathbb{E}[Z_i] = \mathbb{E}[W'_i]. \tag{5.1.1}$$

The reduction theorem only yields an expression for the mean end-to-end delay of type i packets (called the mean type i end-to-end delay), while we are in particular interested in the mean end-to-end delay of type i,j packets (called the mean type i,j end-to-end delay). Even so, the reduction theorem will prove vital for the analysis of the mean type i,j end-to-end delay. In order to apply the reduction to all possible subtrees, we assume that *all* nodes use HoL-based service disciplines.

Remark 5.1.1. The latter assumption can be slightly weakened. We apply the reduction theorem to all node n subtrees, except for nodes n of which all queues store packets arriving directly from the exterior. If one or more queues of node n store packets coming from another node, the service discipline of node n has to be HoL-based. If all queues store packets arriving directly from the exterior, the service discipline can be an arbitrary work-conserving one.

This chapter is organised as follows: In Section 5.2, we derive the mean *overall* (i.e., averaged over all types) end-to-end delay exactly and obtain the approximation of the mean type i,j delay. One of the key steps in this approximation is the assumption that all streams passing through a certain queue at a node have the same mean waiting time in that node. The accuracy of this approximation is numerically analysed using simulation in Section 5.3.

We express the mean type i,j end-to-end delay in terms of the mean waiting time per queue in single-station polling systems. In Section 5.4 we apply the mean end-to-end delay approximation to a model of a network on chip with four switches, where the necessary single-station results are obtained using a known approximation, namely that of Boxma and Meister [34]. For trees satisfying certain symmetry conditions, the mean end-to-end delay approximation becomes exact, as is discussed in Section 5.5. Finally, we present our conclusions in Section 5.6.

5.2 Analysis of the tree

In this section we describe how the reduction theorem can be applied to obtain expressions for the mean end-to-end delay. First, we obtain an exact expression for the mean end-to-end delay of packets of *any* type, called the mean overall end-to-end delay. Second, we approximate the mean type i,j end-to-end delay using the results for the mean overall end-to-end delay.

5.2.1 Overall end-to-end delay

We recall that the reduction theorem states that

$$\mathbb{E}[Z_i] = \mathbb{E}[W_i'],$$

where $\mathbb{E}[Z_i]$ is the mean type i end-to-end delay, and $\mathbb{E}[W_i']$ is the mean waiting time in queue i of the reduced system, which is a polling system with arrivals X_i to queue i, $i = 1, \dots, N$. Because an arbitrary packet is of type i with probability ρ_i/ρ, it follows that $\mathbb{E}[Z]$, the mean overall end-to-end delay, is given by

$$\mathbb{E}[Z] = \sum_{i=1}^{N} \frac{\rho_i}{\rho}\,\mathbb{E}[Z_i] = \sum_{i=1}^{N} \frac{\rho_i}{\rho}\,\mathbb{E}[W_i']. \tag{5.2.1}$$

The right hand side of (5.2.1) can be recognised as part of the conservation law for polling systems [24,32,37]. This law states that for any work-conserving service discipline,

$$\sum_{i=1}^{N} \frac{\rho_i}{\rho}\,\mathbb{E}[W_i'] = C. \tag{5.2.2}$$

The constant C is given by Expression (14) in [24] after division by ρ. For unit packet sizes,

$$\begin{aligned} C &= \frac{1}{2\rho(1-\rho)}\left[\sum_i \mathbb{E}[X_i(X_i-1)] + \sum_i \sum_{j\neq i} \rho_i\rho_j\right] \\ &= -\frac{1}{2} + \frac{1}{2\rho(1-\rho)} \sum_i \mathrm{Var}(X_i). \end{aligned} \tag{5.2.3}$$

By combining this with (5.2.1) and (5.2.2), and using that $X_i = \sum_j X_{i,j}$, with $X_{i,j}$ mutually independent, we obtain

$$\begin{aligned} \mathbb{E}[Z] &= -\frac{1}{2} + \frac{1}{2\rho(1-\rho)} \sum_i \mathrm{Var}(X_i) \\ &= -\frac{1}{2} + \frac{1}{2\rho(1-\rho)} \sum_i \sum_j \mathrm{Var}(X_{i,j}) \end{aligned} \tag{5.2.4}$$

as the mean overall end-to-end delay.

Remark 5.2.1. Equation (5.2.4) gives the mean overall end-to-end delay, regardless of the precise HoL-based service discipline. The work of Morrison [104] and Shalmon [126] entails that Equation (5.2.4) holds without the assumption of HoL-based service disciplines; any work-conserving service discipline suffices. The assumption of HoL-based service disciplines will, however, become crucial in the next subsection. Shalmon [126] also gives an expression for the mean overall end-to-end delay in a concentrating tree network with Poisson arrivals, which is valid in discrete as well as continuous time (Eq. (5.2.4) with $\mathrm{Var}(X_{i,j}) = \rho_{i,j}$).

5.2.2 End-to-end delay per type

In this subsection, we derive an approximation of the mean type i,j end-to-end delay. We express the mean type i,j end-to-end delay in the *network* in the mean waiting time per queue of *single-station* polling systems. For many single-station polling systems, the mean waiting time per queue has been analysed, either in exact form or through an approximation (see [133, 135, 142] for overviews). Moreover, in Chapter 6, we derive a new approximation that can be used to obtain single-station results for a large subclass of HoL-based service disciplines, namely that of Bernoulli service combined with Markovian routing.

The first observation is that the type i,j end-to-end delay consists of the sum of the waiting times of type i,j packets at all nodes along their path from the source to node 0. In other words, if we approximate the mean waiting time of type i,j packets at an arbitrary node, an approximation of the mean type i,j end-to-end delay automatically follows by summing the mean waiting time approximations at the individual nodes.

A second observation is the following: Consider Figure 5.3 and suppose for a moment that we want to determine the mean waiting time of type i,j packets in node i. Everything that happens outside the node i subtree (marked by the dashes) has no influence on the mean waiting time in node i, so it suffices to consider only the node i subtree. Node i, however, is itself the sink of the node i subtree. In order to approximate the mean waiting time of type i,j packets in an arbitrary node, it hence suffices to determine the mean waiting time in the *last* node of an arbitrary network.

In the sequel, we approximate the mean waiting time of type i,j packets in node 0, which leads to an approximation of the mean type i,j end-to-end delay as described by the two observations above. We denote the mean waiting time of type i,j packets in node 0 by $\mathbb{E}[W_{i,j}^{(0)}]$.

It is not immediately clear, however, how $\mathbb{E}[W_{i,j}^{(0)}]$ can be determined: First, it is unclear which of the type i packets in node 0 are actually type i,j packets. The type i,j packets are mixed with packets of type i,j_1, i,j_2, etc. Packets are stored in node 0 in an intricate unknown order that is determined by the service disciplines of the nodes upstream. Second, $\mathbb{E}[W_{i,j}^{(0)}]$ represents the mean waiting time in a polling model where the arrivals are given by the output of the node upstream.

The first difficulty is circumvented by the following approximation:

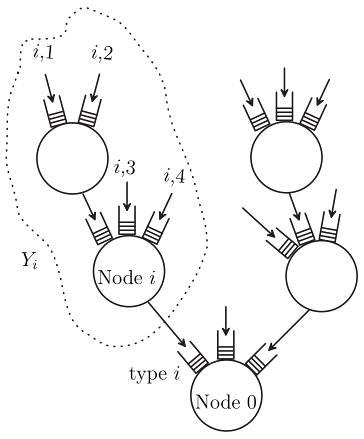


Figure 5.3: The example network.

Approximation 5.2.2. *For every node, the mean waiting time of type i,j packets is equated to the mean waiting time of* all *packets passing through the* same queue *in that node.*

Applying Approximation 5.2.2 to node 0 entails that we approximate the mean waiting time in node 0 of type i,j packets by the mean waiting time of type i packets in node 0, i.e.,

$$\mathbb{E}[W_{i,j}^{(0)}] \approx \mathbb{E}[W_i^{(0)}]. \tag{5.2.5}$$

The quantity $\mathbb{E}[W_i^{(0)}]$, however, still represents the mean waiting time in a polling model where arrivals are given by the output of the node upstream.

We can now circumvent the second difficulty with the reduction theorem: The mean waiting time of type i packets at node 0, $\mathbb{E}[W_i^{(0)}]$, is equal to the mean type i end-to-end delay in the entire tree, $\mathbb{E}[Z_i]$, minus the mean type i end-to-end delay in the node i subtree, denoted by $\mathbb{E}[Y_i]$. Using the reduction theorem (Equation (5.1.1)) we thus obtain

$$\mathbb{E}[W_i^{(0)}] = \mathbb{E}[Z_i] - \mathbb{E}[Y_i] = \mathbb{E}[W_i'] - \mathbb{E}[Y_i].$$

Because all packets in the node i subtree are type i packets, $\mathbb{E}[Y_i]$ is the mean *overall* end-to-end delay in the node i subtree. It follows from the analysis of Section 5.2.1 (i.e., Equation (5.2.4) applied to the node i subtree) that

$$\mathbb{E}[Y_i] = -\frac{1}{2} + \frac{1}{2\rho_i(1-\rho_i)} \sum_j \mathrm{Var}(X_{i,j}). \tag{5.2.6}$$

In summary,

$$\mathbb{E}[W_{i,j}^{(0)}] \approx \mathbb{E}[W_i^{(0)}] = \mathbb{E}[W_i'] - \mathbb{E}[Y_i] \tag{5.2.7}$$

where $\mathbb{E}[Y_i]$ is given by (5.2.6), and $\mathbb{E}[W_i']$ is the mean waiting time in queue i of the reduced system. The two key steps in the derivation of (5.2.7) are Approximation 5.2.2 and application of the reduction theorem.

Remark 5.2.3. If type i,j packets arrive to node 0 directly, there is of course no suitable subtree. In this case, we can replace $\mathbb{E}[Y_i]$ by 0, and $\mathbb{E}[W_{i,j}^{(0)}] \approx \mathbb{E}[W_i^{(0)}] = \mathbb{E}[W_i']$.

5.3 Accuracy of Approximation 5.2.2

In this section, we analyse the accuracy of Approximation 5.2.2 by means of a simulation study over a large parameter space. We consider the smallest non-trivial polling tree network, which consists of two nodes, node 0 and 1, both with two queues (see Fig. 5.4). Queue 1 of node 0 stores packets arriving from node 1 while queue 2 of node 0 stores packets arriving from the exterior directly. There are three different types of packets, namely type 1,1, type 1,2, and type 2,1. All arrivals occur according to ordinary (non-batch) Bernoulli arrival processes, i.e., each time slot an arrival of type i,j takes place with probability $\rho_{i,j}$. We introduce a unit sum weight vector $\nu = (\nu_{1,1}, \nu_{1,2}, \nu_{2,1})$ such that $\rho_{i,j} = \nu_{i,j}\rho$ for a single load parameter ρ. We assume each node uses the 1-limited service discipline.

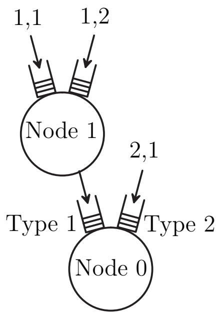


Figure 5.4: The network of Section 5.3.

Without loss of generality, we assume $\nu_{1,1} \leq \nu_{1,2}$. We cover all possible cases of $\nu_{i,j}$ with a stepsize of 0.05 between consecutive values of $\nu_{i,j}$. This leads to a total of 90 possible cases (see Table 5.1). For each case, we run simulations for ρ from 0.01 to 0.99, in steps of 0.01.

We analyse the error made in the approximation of the mean waiting time at node 0 (Eq. (5.2.5)), i.e., we analyse the value of

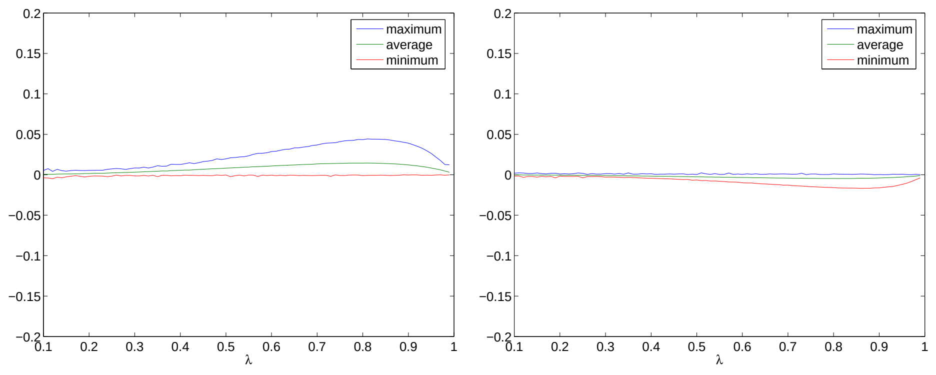
$$\varepsilon_j = \frac{\mathbb{E}[W_1^{(0)}]}{\mathbb{E}[W_{1,j}^{(0)}]} - 1, \qquad j = 1, 2,$$

where both $\mathbb{E}[W_1^{(0)}]$ and $\mathbb{E}[W_{1,j}^{(0)}]$ are determined by simulation.

Figure 5.5 displays the average and extreme values of ε_j over all cases for $j = 1, 2$. It clearly shows that the average error is within a few percent for all loads above 0.1. For loads close to 0, ε_j is the ratio of two numbers close to zero, which leads to

Case	$\nu_{1,1}$	$\nu_{1,2}$	$\nu_{2,1}$	Case	$\nu_{1,1}$	$\nu_{1,2}$	$\nu_{2,1}$
1	0.05	0.05	0.90	35	0.15	0.15	0.70
2	0.05	0.10	0.85	36	0.15	0.20	0.65
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
18	0.05	0.90	0.05	48	0.15	0.80	0.05
19	0.10	0.10	0.80	49	0.20	0.20	0.60
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
34	0.10	0.85	0.05	89	0.45	0.45	0.10
				90	0.45	0.50	0.05

Table 5.1: The 90 cases considered.



(a) $j = 1$ **(b) $j = 2$**

Figure 5.5: Average and extreme values of ε_j over all cases.

some irrelevant variability in the graph. The results for $\rho < 0.1$ have therefore been omitted from the graph.

Apart from average and extreme values of ε_j, it is interesting to see which cases typically induce large errors. Table 5.2 shows the five cases that most frequently have large errors; clearly, cases with large errors are typically quite asymmetric. Additional simulations have furthermore shown that the error is typically largest if such an asymmetric case is combined with a load of around 0.7, 0.8.

There are, however, even more asymmetric cases, which are not in Table 5.2. Apparently, the error is again smaller for very asymmetric cases. To study this effect in more detail, we perform the following experiment: We fix $\rho = 0.8$ and $\nu_{2,1} = 0.1$ (these settings generally lead to larger errors, so that the effect of asymmetry is clearly visible). We vary $\nu_{1,1}$ and $\nu_{1,2}$ subject to the constraints that $\nu_{1,1}+\nu_{1,2} = 0.9$ and $\nu_{1,1} \leq \nu_{1,2}$.

Figure 5.6 shows the values of ε_j, $j = 1, 2$, in this experiment. On the horizontal axis we have $\nu_{1,2} - \nu_{1,1}$, which is a measure for how asymmetric node 1 is: The left

$\nu_{1,1}$	$\nu_{1,2}$	$\nu_{2,1}$
0.15	0.80	0.05
0.15	0.75	0.10
0.20	0.70	0.10
0.20	0.75	0.05
0.25	0.70	0.05

(a) Type 1,1

$\nu_{1,1}$	$\nu_{1,2}$	$\nu_{2,1}$
0.25	0.65	0.10
0.25	0.70	0.05
0.30	0.60	0.10
0.30	0.65	0.05
0.35	0.60	0.05

(b) Type 1,2

Table 5.2: Cases that frequently have larger errors.

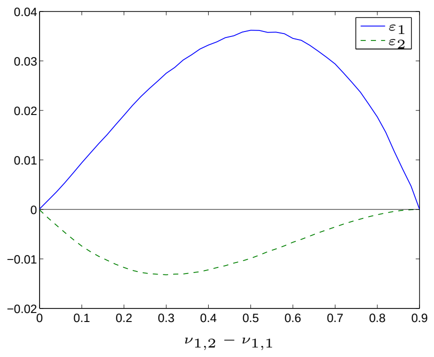


Figure 5.6: The influence of asymmetry.

side corresponds to a symmetric system ($\nu_{1,1} = \nu_{1,2}$), and the right side corresponds to a completely asymmetric system ($\nu_{1,2} - \nu_{1,1} = 0.9$, i.e., $\nu_{1,1} = 0$ and $\nu_{1,2} = 0.9$).

Clearly, the absolute values of the errors increase if node 1 becomes more asymmetric, but only up to a certain point. After this point, the absolute values of the errors decrease again.

5.4 Application to networks on chips

In this section, we show how the approximation of the end-to-end delay can be applied to networks on chips. The required single-station results are obtained using a known approximation for 1-limited polling stations, namely the Boxma-Meister approximation [34]. In Section 5.4.1, we describe the network in more detail, and we analyse the network in Section 5.4.2. In Section 5.4.3 we perform a numerical study of the accuracy of the combination of the mean delay approximation and the Boxma-Meister approximation.

5.4.1 Description

We consider a model of a network on chip with multiple switches (nodes), where all traffic has the same destination (see Figure 5.7). The switches in this network are organised in a mesh topology; all switches have four queues and are placed on a lattice with connections in four directions (up, down, right, left). The routing mechanism of this network is XY-routing, which means that packets first traverse the X-direction, as far as they have to go, and then move in the Y-direction to their destination. There is for example a link between node 3 and node 1, but it is never used because all traffic is headed to node 0. It is thus the particular routing strategy that ensures the mesh topology is a tree network corresponding to the setting of this chapter.

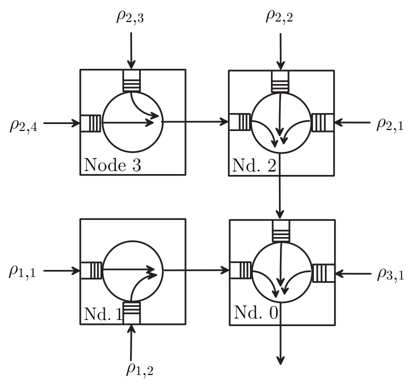


Figure 5.7: A mesh network with 4 switches and 7 input streams

We assume that packets consist of K flits, with K fixed. Recall that a flit is the amount of data that can be transmitted in one time slot. We assume packets arrive according to ordinary (non-batch) Bernoulli arrival processes. Essentially, this amounts to assuming that K units of data arrive according to a Bernoulli process, so we have:

$$X_{i,j} = \begin{cases} 0, & \text{w.p. } 1 - \lambda_{i,j}, \\ K, & \text{w.p. } \lambda_{i,j}, \end{cases}$$

so that $\mathbb{E}[X_{i,j}] = K\lambda_{i,j}$ and $\text{Var}(X_{i,j}) = K^2\lambda_{i,j}(1 - \lambda_{i,j})$. From $\rho_{i,j} = \mathbb{E}[X_{i,j}]$, it follows that $\lambda_{i,j} = \rho_{i,j}/K$.

Inside the switches, packets are served according to round-robin scheduling, which corresponds to the cyclic K-limited service discipline. Furthermore, we assume the switches employ wormhole routing, which has two implications: First, once the first flit of a packet starts transmission at a certain node, the entire packet has to complete transmission before another packet may start transmission. Second, multiple flits of a single packet might be spread out over several nodes. As a result, a size K packet that never has to wait completes transmission over L nodes in $L + K - 1$ time slots, rather than LK in networks without wormhole routing.

5.4.2 Analysis

In this subsection, we derive the end-to-end delays of an arbitrary flit, as well as a *header* (the first flit in a packet). The reduced system, like node 0, uses K-limited service. Because K is fixed, the mean waiting time of a header in this K-limited system is equal to the mean waiting time in a *1-limited* system with service times K and *non-batch* Bernoulli arrival processes with parameter $\lambda_{i,j}$.

We use the Boxma-Meister approximation for the mean waiting time in the latter 1-limited polling system. This approximation states that

$$\mathbb{E}[W'_{i,h}] \approx \frac{1-\rho+\rho_i}{1-\rho+\frac{1}{\rho}\sum_j \rho_j^2} C_K, \tag{5.4.1}$$

where $W'_{i,h}$ is the waiting time of a header in queue i of the reduced system, and C_K is the conservation law constant (cf. (5.2.3)). After dividing Equation (14) in [24] by ρ and some rewriting, we have:

$$C_K = \frac{\rho}{2(1-\rho)} \left(K - \sum_i \sum_j \left(\frac{\rho_{i,j}}{\rho} \right)^2 \right). \tag{5.4.2}$$

The mean waiting time of an *arbitrary* flit in the reduced system, $\mathbb{E}[W'_i]$, is equal to that of a header plus $(K-1)/2$, i.e.,

$$\mathbb{E}[W'_i] = \mathbb{E}[W'_{i,h}] + \frac{K-1}{2}. \tag{5.4.3}$$

Suppose now that $i = 1, 2$ (the case $i = 3$ is slightly different and will be dealt with later). Type i,j packets arrive to node i from the exterior directly and are transmitted towards node 0 flit-by-flit, due to the wormhole routing. The header thus always arrives at node 0 one time slot earlier than the second flit, and always leaves one time slot earlier. The mean waiting time of a header is therefore equal to the mean waiting time of an arbitrary flit, i.e., $\mathbb{E}[W^{(0)}_{i,j,h}] = \mathbb{E}[W^{(0)}_{i,j}]$. We obtain (cf. Eq. (5.2.7)):

$$\mathbb{E}[W^{(0)}_{i,j,h}] = \mathbb{E}[W^{(0)}_{i,j}] \approx \mathbb{E}[W^{(0)}_i] = \mathbb{E}[W'_i] - \mathbb{E}[Y_i],$$

where $\mathbb{E}[Y_i]$ is given by (5.2.6), and $\mathbb{E}[W'_i]$ by (5.4.3).

As in Section 5.2, the mean waiting times of type 1 and 2 packets in the other nodes can be obtained similarly, resulting in an approximation of the mean end-to-end delay.

Type 3 packets arrive at node 0 directly, so $\mathbb{E}[Y_i] = 0$ (see Remark 5.2.3). Furthermore, all type 3 packets are type 3,1 packets and the delay of these packets consists of only the waiting time at node 0. The mean end-to-end delay of an *arbitrary* type 3,1 flit is thus given by:

$$\mathbb{E}[W^{(0)}_{3,1}] = \mathbb{E}[W'_3].$$

Type 3 *headers*, however, spend on average $(K-1)/2$ time slots less in node 0 than an arbitrary flit. The mean end-to-end delay of type 3 headers is hence equal to

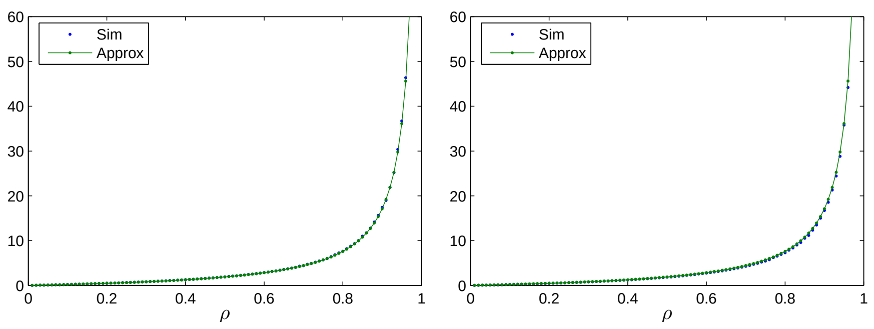
$$\mathbb{E}[W^{(0)}_{3,1,h}] = \mathbb{E}[W'_3] - \frac{K-1}{2} = \mathbb{E}[W'_{3,h}].$$

5.4.3 Numerical results

In this subsection we study the accuracy of the mean end-to-end delay approximation for the following two cases: Balanced load division and homogeneous load division. We again assume there is a unit sum weight vector ν describing the division of the total load ρ over the various input streams, i.e., $\rho_{i,j} = \nu_{i,j}\rho$.

Case I: Balanced load division

By balanced load division we mean that the loads are divided in such a way that at each node all queues receive the same load. That is, we assume $\nu_{3,1} = 1/3$, $\nu_{1,1} = \nu_{1,2} = 1/6$, $\nu_{2,1} = \nu_{2,2} = 1/9$, and $\nu_{2,3} = \nu_{2,4} = 1/18$. In Figure 5.8, we show the most and least accurate of the approximations, i.e., the approximations of the mean end-to-end delay of type 1,1 packets and type 3,1 packets.



(a) $\boldsymbol{i = 1}$, $\boldsymbol{j = 1}$ (most accurate) **(b)** $\boldsymbol{i = 3}$, $\boldsymbol{j = 1}$ (least accurate)

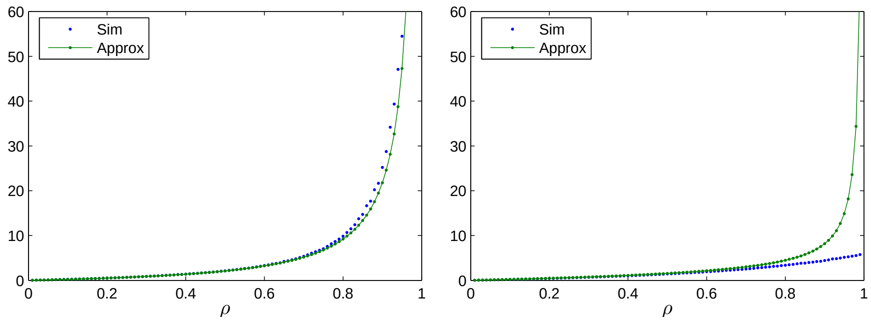
Figure 5.8: The most and least accurate approximations of the mean type i,j end-to-end delay for the balanced load division.

It is clear that the approximation of the mean end-to-end delay is very accurate in this case. This is not very surprising, as all nodes are almost symmetric. For instance, if we apply the reduction theorem, we obtain a polling system with three queues, each with load $\rho/3$. One of these queues has an arrival process that is the superposition of four arrival processes (namely $\sum_j X_{2,j}$), one arrival process is a superposition of two ($\sum_j X_{1,j}$), and one is not a superposition (or a superposition of one). In other words, the loads to all queues are identical, but the arrival processes are superpositions of different Bernoulli arrival processes.

Other than this difference, the system is symmetric, in which case the Boxma-Meister approximation is exact [34]. It is indeed unlikely that such a small asymmetry leads to large errors. Furthermore, we already saw in Section 5.3 that Approximation 5.2.2 is very accurate if the individual nodes are nearly symmetric.

Case II: Homogeneous load division

With the homogeneous load division, all input streams receive a fraction 1/7 of the total load, i.e., $\nu_{i,j} = 1/7$. Again, we show the most and least accurate



(a) $\boldsymbol{i = 2}$, $\boldsymbol{j = 1}$ (most accurate) **(b)** $\boldsymbol{i = 3}$, $\boldsymbol{j = 1}$ (least accurate)

Figure 5.9: The most and least accurate approximations of the mean type i,j end-to-end delay for the homogeneous load division.

approximation (see Figure 5.9).

The approximations are very accurate up to a load of roughly 0.7. Beyond this load, the approximation is only accurate for the input stream with the highest load. This can be explained by the fact that node 0 is rather asymmetric. After all, one queue receives a fraction of 4/7 of the total load, while the other queues get fractions 2/7 and 1/7.

The accuracy of the Boxma-Meister approximation degrades for heavily loaded very asymmetric systems [34]. If the 1-limited service discipline is used, all queues receive a positive fraction of the capacity of the server, even if the load is larger than 1. Some queues may thus remain stable even though others become unstable. The conservation law constant C_K, however, tends to infinity. As a result, the mean waiting time approximations tend to infinity for *all* queues, including those that are still stable and have a finite waiting time for $\rho = 1$. We observe this phenomenon in Figure 5.9 too: The mean end-to-end delay approximation is unbounded, whereas the simulated mean delay is still finite if the load is 1.

Other approximations have been suggested, that might perform better in some cases (for instance, Blanc [28], Groenendijk and Levy [68], Srinivasan [128], and Van Vuuren and Winands [140]). Depending on the desired properties of the approximation (e.g., closed-form or numerical procedure) and the characteristics of the tree (e.g., heavily or not heavily loaded, very asymmetric or roughly symmetric) one has to choose which single-station approximation to use in order to obtain the best results. For example, the Boxma-Meister approximation is a closed-form expression, but its accuracy degrades for heavily loaded asymmetric systems. The new approximation of Chapter 6 is generally better in this case, but it is numerically intensive.

Remark 5.4.1. Boxma and Meister propose a refinement to their procedure in [34]. This refinement significantly improves the accuracy of the approximation for heavy loads, but also destroys the closed-form property. Furthermore, Boxma and Meister suggest in [33] that a group of heavily loaded queues can be replaced by a suitable switch-over time, which also improves the accuracy of the approximation for heavily loaded asymmetric systems.

5.5 Exact results

In this section, we consider networks for which the mean type i,j end-to-end delay approximation is, in fact, exact rather than an approximation. The mean type i,j end-to-end delay approximation is exact if the following two requirements are met: First, in the approximation of the end-to-end delay we reduce several subtrees to single-station polling systems. For all these systems, the mean waiting time per queue must be known exactly. Second, Approximation 5.2.2 has to be exact; all streams passing through the same queue of any node must have the same mean waiting time in that node.

In Section 1.4.2, we mentioned that single-station polling systems with a service discipline satisfying the branching property, such as gated service and exhaustive service, can be analysed exactly. Gated service, however, is not HoL-based, which means that the network cannot be reduced to a single-station polling system using the reduction theorem. Exhaustive service, on the other hand, is HoL-based. The mean queue lengths in a polling station with exhaustive service can be found in [124] and [132], where they are given in terms of a solution to a system of equations. Although these expressions are still implicit, the necessary single-station results can be determined numerically from them.

Besides polling stations with a branching-type service discipline, the mean waiting time per queue can also be obtained for special cases such as 2-queue systems and symmetric systems. We introduce symmetry conditions for single stations with a HoL-based service discipline, and obtain the mean waiting time per queue in such systems in Section 5.5.1.

The second requirement for an exact analysis of the mean type i,j delay, namely that Approximation 5.2.2 has to be exact, is also closely related to symmetry; Approximation 5.2.2 indeed becomes an exact statement if the entire network satisfies certain symmetry conditions. This is explored in Section 5.5.2.

5.5.1 Symmetric stations

We introduce the concept of *symmetric service disciplines*. A service discipline is symmetric if it satisfies the following three properties: First, at each queue the server serves a number of packets according to a fixed rule such as 1-limited, exhaustive, or Bernoulli service. This rule is the same for all queues. Second, Markovian routing is used, which means that after service of queue j, the server moves to queue $k \neq j$

with probability p_{jk}. Third, the routing matrix $P = (p_{jk})$ is circulant, i.e.,

$$P = \begin{pmatrix} 0 & p_2 & p_3 & \dots & p_N \\ p_N & 0 & p_2 & \dots & p_{N-1} \\ p_{N-1} & p_N & 0 & \dots & p_{N-2} \\ \vdots & \vdots & & \ddots & \vdots \\ p_2 & p_3 & p_4 & \dots & 0 \end{pmatrix}, \tag{5.5.1}$$

or can be written in circulant form after a permutation of the queues. Note that cyclic routing has a circulant P-matrix with $p_2 = 1$.

If a polling system uses a symmetric service discipline and has stochastically identical arrival processes, we call it *symmetric*. In a symmetric polling system, the mean waiting time is the same for all queues. After all, with a circulant P-matrix, all rows of P are identical, apart from being shifted. In addition, all arrival processes are stochastically identical, so there is no difference between the various queues. In other words, the mean waiting times per queue are invariant under permutation of the queues.

Because the mean waiting times at all queues are the same, the mean waiting times can be obtained using the conservation law (5.2.2). For example, if the reduced system is symmetric, $\mathbb{E}[W_i']$ is exactly given by

$$\mathbb{E}[W_i'] = C = -\frac{1}{2} + \frac{1}{2\rho(1-\rho)} N \mathrm{Var}(X_1),$$

for all i. This expression holds regardless of the precise service discipline, as long as it is symmetric.

5.5.2 Symmetric trees

Approximation 5.2.2 states that all packets passing through the same queue at a node are assumed to have the same mean waiting time at that node. In this section, we introduce a class of trees for which Approximation 5.2.2 is in fact not an approximation but an exact statement.

We say that a polling tree is symmetric if it satisfies all of the following four properties:

1. All external arrival processes are stochastically identical.
2. All external arrivals occur at the same level. Here, the level of a node is defined as the distance to the sink (see Fig. 5.10).
3. All nodes within a particular level have the same number of queues.
4. All nodes within a particular level use the same symmetric service discipline.

Consider an arbitrary polling tree network, let node i be the node directly above queue i of node 0, and suppose that the *node i subtree* is symmetric. An example of such a tree is displayed in Figure 5.10.

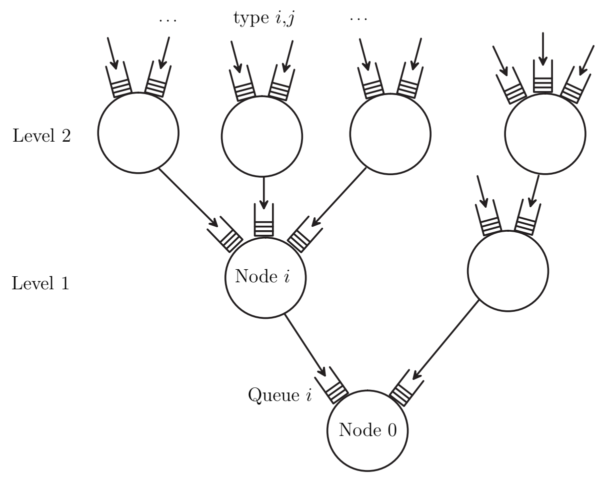


Figure 5.10: A tree with a symmetric node i subtree

Since the node i subtree is symmetric, there is no distinction between the type i,j packets, $j = 1, \dots, N_i$. In particular, the mean waiting time of type i,j packets at node 0 is invariant under permutation of j:

$$\mathbb{E}[W_{i,j}^{(0)}] = \mathbb{E}[W_i^{(0)}].$$

Approximation 5.2.2 is thus an exact statement for node 0. Moreover, because all subtrees within the node i subtree are again symmetric, Approximation 5.2.2 is exact for *all* nodes in the node i subtree. Note that there are no conditions on nodes outside the node i subtree; all conditions apply to the node i subtree, and all other nodes (including node 0) are arbitrary.

Moreover, because the mean waiting time of type i,j packets is invariant under permutation of j for all nodes, the mean end-to-end delay of type i,j packets is invariant under permutation of j as well. We thus obtain:

$$\mathbb{E}[Z_{i,j}] = \mathbb{E}[Z_i] = \mathbb{E}[W_i'].$$

If exact results are available for $\mathbb{E}[W_i']$ (for instance if the reduced system is symmetric or if it uses exhaustive service), the mean type i,j end-to-end delay can thus be obtained exactly using a *single* invocation of the reduction theorem.

5.6 Conclusion

If all nodes in a concentrating tree network use a HoL-based service discipline, the mean type i,j end-to-end delay can be analysed by repeated application of the reduction theorem (Theorem 4.2.6) in combination with the approximation assumption that packets from different sources passing through the same queue of any node have the same mean waiting time at that node (Approximation 5.2.2). Through this analysis, the mean type i,j end-to-end delay is expressed in the mean waiting time per queue in single-station polling systems.

For the 1-limited service discipline, Approximation 5.2.2 is very accurate over the entire parameter space of the smallest non-trivial tree. It is especially accurate for nearly symmetric systems and extremely asymmetric systems, and somewhat less accurate for moderately asymmetric systems.

We applied the mean end-to-end delay approximation to a model of a network on chip with four switches. The necessary single-station results were obtained using the Boxma-Meister approximation. Although the Boxma-Meister approximation is less accurate for asymmetric systems, it can still be used to accurately approximate the mean type i,j end-to-end delay up to moderately high loads (around 0.7), even in rather asymmetric networks. If the mean end-to-end delay approximation is applied to specific trees, one has to choose which single-station approximation to use based on the characteristics of the tree (e.g., nearly symmetric, very asymmetric, etc.) in order to obtain the most accurate results.

In the special case that the subtree directly above queue i is symmetric, Approximation 5.2.2 becomes an exact statement rather than an approximation. If, in addition, exact results are available for the reduced system (for example if that station is symmetric too, or if it uses the exhaustive service discipline), the mean end-to-end delay per source can be determined exactly.

Chapter 6

Polling systems with Bernoulli service and Markovian routing

In Chapter 4, we established the reduction theorem, which states that concentrating tree networks of polling systems can be reduced to single-station polling systems, under the assumption that node 0 uses a HoL-based service discipline. In Chapter 5, we used the reduction theorem to derive an approximation of end-to-end delays in such networks, provided all nodes use a HoL-based service discipline.

In the approximation of Chapter 5, we expressed the mean end-to-end delay per source in terms of single-station results. Single-station results for *general* HoL-based service disciplines, however, are not known. In this chapter, we derive a new single-station approximation for a large subclass of HoL-based service disciplines: Bernoulli scheduling combined with Markovian routing.

Rather than only approximating mean waiting times, we approximate the marginal queue length distribution in polling systems with Bernoulli service and Markovian routing. The key step of our approximation is the translation of the polling system to a structured Markov chain, while truncating all but one queue. Numerical experiments show that the approximation is very accurate in general.

6.1 Background

We devise an approximation of the queue length distribution of a discrete-time polling system with batch arrivals, fixed packet sizes, Bernoulli service, and Markovian routing. Bernoulli service means that after service of a packet from queue i, the server serves queue i again with probability $q(i)$ and moves to another queue with probability $1 - q(i)$. Markovian routing means that if the server moves to another queue, it moves to queue j with probability $P(i, j)$ for $j \neq i$, independently of everything else.

The essential part of our approximation is the translation of the polling system to a Structured Markov Chain (SMC) of the $M/G/1$ type (see [106]). An SMC of $M/G/1$ type is a Markov chain of which the states can be described as tuples (l, ϕ), where l, called the level, is an element from $\{0, 1, \ldots\}$ and ϕ, called the phase, an element from some finite set (the phase space). The Markov chain has a transition probability matrix of the following block-partitioned form (hence the name structured):

$$\begin{pmatrix} B_0 & B_1 & B_2 & B_3 & \dots \\ C_0 & A_1 & A_2 & A_3 & \dots \\ 0 & A_0 & A_1 & A_2 & \dots \\ 0 & 0 & A_0 & A_1 & \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{6.1.1}$$

The matrix A_0 describes transitions where the level decreases by one, A_1 describes transitions within a level, and so on. The behaviour of the Markov chain at the boundary $l = 0$ may be different from that in the interior $l > 0$, which is reflected by the matrices B_k, $k = 0, 1, \ldots$, and C_0.

The idea behind our approximation is simple yet very effective: Instead of analysing the queue contents of *all* queues at the same time, we focus on the precise contents of *one* queue. For the other queues we only keep track of whether there are $0, 1, \ldots, B - 1$, or 'B or more' packets in the queues. By also keeping track of the index of the queue that is being served, we obtain an SMC of the $M/G/1$ type.

The truncation of queue lengths implies that we have to introduce additional parameters representing the probability that the contents of a queue go from 'B or more' to $B - 1$. The values of these unknown parameters are determined iteratively. In each iteration the equilibrium distribution of the SMCs is used to update the values of these parameters, until the values have converged. As $B \to \infty$, our approximation becomes exact, but it turns out that setting $B = 2$ already leads to an accurate approximation.

This chapter is organised as follows: In Section 6.2 we describe the model in more detail and in Section 6.3 we give an overview of the relevant literature. We describe our approximation in Section 6.4. The accuracy of the approximation is studied in Sections 6.5 and 6.6: In Section 6.5 we perform a detailed analysis of a single case, and in Section 6.6 a global analysis of multiple cases.

In the implementation of our approximation, we used Kronecker products to determine the transition probability matrix, which makes the computations significantly faster. Nevertheless, we describe our approximation without the use of Kronecker products in Section 6.4 for the sake of readability. In Section 6.7, we give an alternative expression for the transition probability matrix of the structured Markov chains using Kronecker products. Lastly, we present our conclusions in Section 6.8.

6.2 Model description

We assume that all packets have unit size and that packets arrive to the queues according to independent batch Bernoulli arrival processes. The model for which we derive our approximation is thus a discrete-time polling model with N infinite queues and the following characteristics:

1. Batch Bernoulli arrival processes, i.e., every time slot l arrivals occur to queue k with probability $x_k(l)$.
2. Deterministic service times, equal to 1.
3. Bernoulli service discipline, i.e., after service of queue i the server serves queue i again with probability $q(i)$ and moves to another queue with probability $1 - q(i)$.
4. Markovian routing, i.e., *if* the server moves, it moves from queue i to queue $j \neq i$ with probability $P(i,j)$. We assume $P(i,i) = 0$.
5. Zero switch-over times.

The Bernoulli service discipline and Markovian routing dictate that, after a service completion at queue i, the server moves to queue $j \neq i$ with probability $(1 - q(i))P(i,j)$ and stays at queue i (i.e., $j = i$) with probability $q(i)$. If queue j is empty, but not *all* queues are empty, the server immediately moves to another queue according to the routing matrix P, and again if this queue is empty too, and so on, until it finds a non-empty queue. If *all* queues are empty the server remains at queue j. When new packets arrive to any of the queues, the server again moves according to the routing matrix P, until it moves to one of the non-empty queues. All of these movements happen instantaneously.

Remark 6.2.1. The model described above is not equivalent to a model where the server always moves according to a matrix R, with $R(i,i) = q(i)$ and $R(i,j) = (1 - q(i))P(i,j)$, until it finds a non-empty queue. With the exhaustive service discipline, $R(i,i) = 1$, which means that the server stays at queue i indefinitely, even after it has become empty. Such behaviour cannot occur in our model because $P(i,i) = 0$.

To prevent a situation where the server cannot reach some non-empty queues, we assume P is irreducible. Furthermore, we assume the polling station operates under the late arrival - departures first arrival model (see Section 1.3.2). Arrivals and service completions thus happen at the end of time slots with service completions before arrivals. Packets arriving to an empty queue at the end of time slot $[t-1,t)$ may be served in time slot $[t,t+1)$. Finally, we assume the polling system is stable. Because the switch-over times are zero, the polling system is work-conserving and therefore stable if the total load is less than 1. We thus assume that $\sum_{k,l} l x_k(l) < 1$.

6.3 Relevant literature

In this section, we review the literature relevant to our study of polling systems with Bernoulli service and Markovian routing. The Bernoulli service discipline was introduced by Keilson and Servi [78] in a *single-queue* vacation system where after each service completion the server takes a vacation with probability p. Tedijanto [137] later analysed a *multi-queue* polling system with the Bernoulli service discipline.

Markovian routing was analysed by Boxma and Weststrate in [36]. They derive the pseudo-conservation law for Markovian routing combined with the traditional service disciplines (exhaustive, gated, and 1-limited), see also Weststrate [143]. Independently of Boxma and Weststrate, Srinivasan [129] considers a similar system, but in a slightly more general setting.

More important to our work, however, are results on queue lengths. Resing [117] established that polling systems satisfying a certain ‘branching property’ can be viewed as branching processes and exact results can be obtained. For polling systems that do not satisfy the branching property, such as systems with the Bernoulli service discipline, even *mean* queue lengths are unknown, except for special cases such as 2-queue and symmetric systems. The 2-queue system with Bernoulli service was analysed by Feng et al. [55], the 2-queue system with 1-limited service without switch-over times by Boxma and Cohen [43] and Eisenberg [52], and with switch-over times by Boxma and Groenendijk [31]. Furthermore, a 2-queue system with switch-over times and a combination of Bernoulli and exhaustive service was studied by Weststrate and Van der Mei [144]. Finally, a symmetric polling system with random polling was analysed by Kleinrock and Levy [84]. Random polling is a special case of Bernoulli service and Markovian routing where, after every service completion, queue j is served with probability $P(j)$.

Since exact results are known only for special cases, and their derivation gives little hope for extensions to more general cases, we focus on approximations instead. In research of polling systems, the continuous-time domain received far more attention than the discrete-time domain. One of the few examples of discrete-time approximations is the approximation by Frigui and Alfa [58], where a polling model with time-limited service is studied.

In the continuous-time domain, the queue length distribution in a polling system with the Bernoulli service discipline and cyclic routing was analysed using the power series approximation of Blanc [26–29]. For the cyclic 1-limited system, the most important special case of our model, other approximations of the queue length distributions exist: First, Van Vuuren and Winands [140] use structured Markov chains to approximate queue lengths in a k_i-limited polling system with cyclic routing. Their approach, however, is very different from ours since they use structured Markov chains to approximate visit and intervisit periods. Second, Leung obtains an approximation for a polling system with the probabilistically limited service discipline (which includes 1-limited) using discrete Fourier transforms [94]. Third, Lee and Sengupta approximate queue length distributions in a polling model with a reservation mechanism using an iterative approximation of visit and intervisit periods [92].

Other authors, such as Boxma and Meister [34], Fuhrmann and Wang [59], Levy and Groenendijk [68], and Srinivasan [128], only approximated *mean* waiting times in cyclic 1-limited polling systems (and by Little's law, mean queue lengths). Of those, we found that the Boxma-Meister [34] and Levy-Groenendijk [68] approximations could be extended to the discrete-time domain without much additional effort. We compare our approximation with these two approximations in Section 6.5.

6.4 The approximation

In this section we describe our approximation in more detail. First, we introduce the phase spaces and derive the transition probability matrices. The iterative determination of the probabilities that the contents of the truncated queues go from 'B or more' to $B-1$ is discussed at the end of this section.

For every queue, we construct an SMC such that the exact contents of that queue are stored in the level. The truncated contents of the other queues, as well as the index of the queue in service (called the service index) are stored in the phase. The SMC where the contents of queue i are stored in the level, is called the 'SMC of queue i'. All SMCs describe the state of the system at integral times t, so immediately before the start of the service of a packet and immediately after arrivals, departures and server movements.

Every phase of the SMC of queue i is described by an N-dimensional vector $(j, n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_N)$, where j is the service index, $n_k = 0, \ldots, B-1$ means there are n_k packets in queue k, and $n_k = B$ means there are B or more packets in queue k, $k \neq i$. The phase space of level $n_i > 0$ consists of all such combinations, except that the service index cannot be j if queue j is empty. We thus obtain

$$\Phi_i = \{1, \ldots, N\} \times \{0, 1, \ldots, B\}^{N-1} \setminus \{(j, n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_N) : n_j = 0 \text{ (for } j \neq i)\}$$

as the phase space for level $n_i > 0$.

The phase space of level $n_i = 0$, is different: First, queue i cannot be served because it is empty. Second, if all queues are empty, the server waits at queue j until new packets arrive to any of the queues. Hence, the phase space of level $n_i = 0$

is

$$\widetilde{\Phi}_i = \{1, \dots, N\} \times \{0, 1, \dots, B\}^{N-1} \setminus \{(j, n_1, \dots, n_{i-1}, n_{i+1}, \dots, n_N) : \ j = i \text{ or } n_j = 0 \ \} \cup \{(j, 0, \dots, 0) : j = 1, \dots, N\},$$

where $(j, 0, \dots, 0)$ means that the server is waiting at queue j until new packets arrive.

Remark 6.4.1. The meaning of phase $(i, 0, \dots, 0)$ depends on whether it is combined with level $n_i > 0$ or $n_i = 0$. If $n_i > 0$, phase $(i, 0, \dots, 0)$ means that a packet from queue i will be served in the next time slot and that all *other* queues are empty. If $n_i = 0$ it means that the *entire* system, including queue i, is empty and the server is waiting at queue i.

In order to describe the transition probability matrix of the SMC of queue i, we divide the movement of the server from one queue to another into two parts: First, the server chooses a queue it *would like* to serve, regardless of whether this queue is empty or not, i.e., the server stays at queue j with probability $q(j)$ and moves to queue k with probability $(1-q(j))P(j,k)$. Second, the server keeps moving according to matrix P until it finds a non-empty queue (provided the server was not already at a non-empty queue, and not all queues are empty).

The transition probability matrix of the SMC of queue i is now given by

$$\begin{pmatrix} B_{i,0}\widetilde{\Psi}_i & B_{i,1}\Psi_i & B_{i,2}\Psi_i & B_{i,3}\Psi_i & \dots \\ A_{i,0}\widetilde{\Psi}_i & A_{i,1}\Psi_i & A_{i,2}\Psi_i & A_{i,3}\Psi_i & \dots \\ 0 & A_{i,0}\Psi_i & A_{i,1}\Psi_i & A_{i,2}\Psi_i & \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{6.4.1}$$

Here, the matrices $A_{i,l}$ and $B_{i,l}$ describe arrivals, departures, and the first part of the server movements. The matrices Ψ_i and $\widetilde{\Psi}_i$ describe the second part of the server movements. Because queue i is empty if and only if the process is in level $n_i = 0$, there are different matrices for level $n_i = 0$ and levels $n_i > 0$, denoted by $\widetilde{\Psi}_i$ and Ψ_i respectively. These matrices are specified in more detail below.

Remark 6.4.2. The matrices $A_{i,l}$ give probabilities of transitions from phases in the phase space Φ_i to *all* possible vectors $(j, n_1, \dots, n_{i-1}, n_{i+1}, \dots, n_N)$ with $j = 1, \dots, N$ and $n_k \in \{0, 1, \dots, B\}$, especially including those where the server is positioned at an *empty* queue. The matrix Ψ_i describes transitions from such vectors to phases in Φ_i, where the server is *not* allowed to be positioned at empty queues. The products of these matrices thus indeed give transition probabilities on the phase space Φ_i.

Determination of Ψ_i and $\widetilde{\Psi}_i$

The matrices Ψ_i and $\widetilde{\Psi}_i$ can be determined as follows: Suppose that the server is positioned at an empty queue, but not all queues are empty. Define I and J as the subsets of empty and non-empty queues, respectively. The matrix P constitutes a Markov chain on $\{1,\dots,N\}$. The probability that the server moves from $i \in I$ to $j \in J$ is equal to the probability that the first visit of that Markov chain to set J occurs at state j, given that the Markov chain starts in state i. The matrices Ψ_i and $\widetilde{\Psi}_i$ follow from computing that probability for all phases (see, e.g., Section 2.11 of Resnick [119] for details).

Determination of $A_{i,l}$ and $B_{i,l}$

In order to identify the contents of $A_{i,l}$ and $B_{i,l}$, $l = 0,1,\dots$, we introduce matrices R describing changes in the service index, X_k describing changes in the contents of queue $k \neq j$, where j is the queue in service, and $Y_{i,j}$ and $\widetilde{Y}_{i,j}$ describing changes in the contents of queue j for level $n_i > 0$ and $n_i = 0$ respectively.

We define $R(j,j')$ as the probability that, after service of queue j, the server moves to queue j':

$$R(j,j') = \begin{cases} (1-q(j))P(j,j'), & \text{if } j \neq j', \\ q(j), & \text{if } j = j'. \end{cases}$$

The matrix X_k is such that $X_k(n_k,n'_k)$ is the probability that the contents of queue k go from n_k to n'_k, with $n_k, n'_k \in \{0,\dots,B\}$ given queue k is not in service. We have:

$$X_k = \begin{pmatrix} x_k(0) & x_k(1) & \dots & x_k(B-1) & 1-\sum_{l<B} x_k(l) \\ 0 & x_k(0) & \dots & x_k(B-2) & 1-\sum_{l<B-1} x_k(l) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & x_k(0) & 1-x_k(0) \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix},$$

where $x_k(l)$ is the probability that l packets arrive to queue k.

We define the matrix $Y_{i,j}$, such that $Y_{i,j}(n_j,n'_j)$ is the probability that, in level $n_i > 0$, queue j goes from n_j to n'_j, with $n_j \in \{1,\dots,B\}$ and $n'_j \in \{0,\dots,B\}$. Here, $n_j \geq 1$ because service of queue j implies that queue j is non-empty.

For queue j, we make the approximation assumption that its contents go from 'B or more' to $B-1$ with a fixed probability denoted by $\widetilde{\zeta}_{i,j}$ and $\zeta_{i,j}$ for level $n_i = 0$ and $n_i > 0$ respectively. The rationale behind this level dependence is that, due to correlation between queue lengths, if queue i is empty it is more likely that the contents of queue j are small, and hence it is also more likely that queue j goes from 'B or more' to $B-1$. The parameters $\zeta_{i,j}$ and $\widetilde{\zeta}_{i,j}$, $j \neq i$, are called the truncation parameters of queue i. The values of these parameters are determined iteratively, as will be described in more detail at the end of this section. For now, we simply assume that they have a certain value. It follows that $Y_{i,j}$ is given by

$$Y_{i,j} = \begin{pmatrix} x_j(0) & \dots & x_j(B-2) & x_j(B-1) & 1-\sum_{l<B} x_j(l) \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & x_j(0) & x_j(1) & 1-\sum_{l<2} x_j(l) \\ 0 & \dots & 0 & \zeta_{i,j}x_j(0) & 1-\zeta_{i,j}x_j(0) \end{pmatrix}.$$

Likewise, $\widetilde{Y}_{i,j}(n_j, n'_j)$ is the probability that the contents of queue j go from n_j to n'_j, for level $n_i = 0$. The matrix $\widetilde{Y}_{i,j}$ is identical to $Y_{i,j}$, except that $\widetilde{\zeta}_{i,j}$ is substituted for $\zeta_{i,j}$.

The matrix $A_{i,l}$ describes changes where queue i goes up by $l-1$ levels. This happens if there are either $l-1$ arrivals and no service completion, or l arrivals and a service completion. It follows that the probability of going from phase $\omega = (j, n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_N)$ to phase $\omega' = (j', n'_1, \ldots, n'_{i-1}, n'_{i+1}, \ldots, n'_N)$ and going up by $l-1$ levels is given by

$$A_{i,l}(\omega, \omega') = x_i(l-1) R(j, j') Y_{i,j}(n_j, n'_j) \prod_{\substack{k \neq j \\ k \neq i}} X_k(n_k, n'_k),$$

for $j \neq i$, with $x_i(-1) := 0$, and

$$A_{i,l}(\omega, \omega') = x_i(l) R(j, j') \prod_{k \neq j} X_k(n_k, n'_k),$$

for $j = i$. Here, the probability that the service index changes from j to j' is given by $R(j, j')$. The probability that the contents of the queues that are not in service change from n_k to n'_k is given by $X_k(n_k, n'_k)$. Finally, the probability that the contents of the queue in service change from n_j to n'_j is $Y_{i,j}(n_j, n'_j)$.

The matrices $B_{i,l}$ are slightly different. First, if all queues are empty, the server remains at the same queue. Second, there is never a service completion at queue i, because queue i is empty. Third, the transitions of the queue in service, queue j, are not given by $Y_{i,j}$ but by $\widetilde{Y}_{i,j}$. We obtain:

$$B_{i,l}(\omega, \omega') = x_i(l) \prod_{k \neq i} X_k(0, n'_k),$$

if $\omega = (j, 0, \ldots, 0)$, and

$$B_{i,l}(\omega, \omega') = x_i(l) R(j, j') \widetilde{Y}_{i,j}(n_j, n'_j) \prod_{\substack{k \neq j \\ k \neq i}} X_k(n_k, n'_k),$$

otherwise.

We have specified all the matrices needed to determine the equilibrium distribution of the SMC of queue i. To compute the equilibrium distribution, we use the software tools of Bini et al. [22, 23]. For more details on how to compute the equilibrium distribution, the reader is referred to [106].

Determination of $\zeta_{i,j}$ and $\widetilde{\zeta}_{i,j}$

In the description of the SMCs, we introduced the truncation parameters of queue i, $\zeta_{i,j}$ and $\widetilde{\zeta}_{i,j}$. We use an iterative procedure to compute the values of these parameters and we denote their value in step m of the iteration by $\zeta_{i,j}^{(m)}$ and $\widetilde{\zeta}_{i,j}^{(m)}$.

In each step of the iterative procedure, we determine new values for the truncation parameters of one queue by means of the most recently computed equilibrium distributions of the other queues: We first determine the values of the truncation parameters of queue 1 and compute the equilibrium distribution of queue 1 with these values. We determine the truncation parameters of queue 2 using the newly computed equilibrium distribution of queue 1, as well as the previous equilibrium distributions of queues $3, 4, \ldots, N$. We then compute the new equilibrium distribution of queue 2, and, with that, new values for the truncation parameters of queue 3, and so on, until the values of all truncation parameters have converged.

The truncation parameters of queue i describe the probability that, without arrivals to queue j, the contents of queue $j \neq i$ go from 'B or more' to $B-1$, given that a service completion occurs at queue j. Without arrivals and with a service completion, a transition from 'B or more' to $B-1$ occurs if the contents of queue j are in fact equal to B. If we denote the length of queue j by Q_j, and the service index by S, the probability of such a transition is thus given by

$$\mathbb{P}(Q_j = B | Q_j \geq B, Q_i > 0, S = j), \qquad j \neq i, \tag{6.4.2a}$$

for level $n_i > 0$ and

$$\mathbb{P}(Q_j = B | Q_j \geq B, Q_i = 0, S = j), \qquad j \neq i, \tag{6.4.2b}$$

for level $n_i = 0$.

We denote the equilibrium distribution of the SMC of queue i in step m of the iteration by $\pi_i^{(m)}(j, n_1, \ldots, n_N)$. By evaluating the conditional probabilities in (6.4.2a) and (6.4.2b) and substituting the corresponding, most recently computed, equilibrium probabilities of the SMC of queue j, we obtain

$$\zeta_{i,j}^{(m)} = \begin{cases} \dfrac{\sum\limits_{n_j=B, n_i\geq 1} \pi_j^{(m)}(j,n_1,\ldots,n_N)}{\sum\limits_{n_j\geq B, n_i\geq 1} \pi_j^{(m)}(j,n_1,\ldots,n_N)}, & \text{for } j < i, \\[2ex] \dfrac{\sum\limits_{n_j=B, n_i\geq 1} \pi_j^{(m-1)}(j,n_1,\ldots,n_N)}{\sum\limits_{n_j\geq B, n_i\geq 1} \pi_j^{(m-1)}(j,n_1,\ldots,n_N)}, & \text{for } j > i, \end{cases} \tag{6.4.3a}$$

and

$$\tilde{\zeta}_{i,j}^{(m)} = \begin{cases} \dfrac{\sum\limits_{n_j=B, n_i=0} \pi_j^{(m)}(j,n_1,\ldots,n_N)}{\sum\limits_{n_j\geq B, n_i=0} \pi_j^{(m)}(j,n_1,\ldots,n_N)}, & \text{for } j < i, \\[2ex] \dfrac{\sum\limits_{n_j=B, n_i=0} \pi_j^{(m-1)}(j,n_1,\ldots,n_N)}{\sum\limits_{n_j\geq B, n_i=0} \pi_j^{(m-1)}(j,n_1,\ldots,n_N)}, & \text{for } j > i, \end{cases} \tag{6.4.3b}$$

as the values for the truncation parameters of queue i in step $m \geq 1$. All sums are taken over $0 \leq n_k \leq B$, for $k \neq i, j$.

For $i < j$, Equations (6.4.3a) and (6.4.3b) express $\zeta_{i,j}^{(1)}$ and $\widetilde{\zeta}_{i,j}^{(1)}$ in terms of $\pi_j^{(0)}(.)$, which is not defined. In this case, we choose

$$\zeta_{i,j}^{(1)} = \widetilde{\zeta}_{i,j}^{(1)} = 1, \qquad \text{for } j > i. \tag{6.4.3c}$$

Numerical experiments indicate that setting the initial values of the truncation parameters to 1 is important. Small values of the truncation parameters indicate that there are many packets in the other queues requiring service. If the initial truncation parameters are (much) smaller than 1, the SMC of queue i might even be transient, though in reality the polling system is recurrent. If the SMC of queue i is transient, its equilibrium distribution cannot be determined and new values for the truncation parameters cannot be found.

If the initial values are chosen equal to 1, the value of the truncation parameters converged in all our examples, though we cannot formally prove convergence. Choosing the initial parameters less than 1 and closer to their limiting values might speed up convergence, at the risk that no convergence occurs at all if the initial values are too small.

Finally, after convergence of the truncation parameters, the approximation of the marginal distribution of the length of queue i follows from the equilibrium distribution of the SMC of queue i. Namely, $\mathbb{P}(Q_i = l)$ is approximated by the sum of the equilibrium probabilities of all phases at level $n_i = l$.

Our algorithm is summarised below:

Algorithm 6.4.3.

```
0. Fix $\varepsilon$ (for instance $\varepsilon = 10^{-8}$) and set $m = 1$.
1. Determine $\Psi_i$ and $\widetilde{\Psi}_i$ for $i = 1, \dots, N$.
2. For $i = 1, \dots, N$:
    (a) Use Equation (6.4.3a), (6.4.3b), or (6.4.3c) to determine $\zeta_{i,j}^{(m)}$ and $\widetilde{\zeta}_{i,j}^{(m)}$.
    (b) Determine $A_{i,l}$ and $B_{i,l}$ with $\zeta_{i,j} = \zeta_{i,j}^{(m)}$ and $\widetilde{\zeta}_{i,j} = \widetilde{\zeta}_{i,j}^{(m)}$.
    (c) Determine the equilibrium distribution of queue $i$, $\pi_i^{(m)}(\cdot)$.
3. Stop if $m \geq 2$ and $\max_{i,j}\{|\zeta_{i,j}^{(m)} - \zeta_{i,j}^{(m-1)}|, |\widetilde{\zeta}_{i,j}^{(m)} - \widetilde{\zeta}_{i,j}^{(m-1)}|\} < \varepsilon$. Otherwise,
   set $m = m + 1$ and repeat step 2.
```

6.5 Numerical results

In this section, we study the accuracy of our approximation for a single case with 4 queues, 1-limited service and cyclic routing. The batch sizes are governed by a Poisson distribution with parameter ρ_i, where $(\rho_1, \dots, \rho_4) = (0.1, 0.2, 0.3, 0.4)\rho$. The approximated queue length distributions are compared with simulation outcomes in Tables 6.1 for $\rho = 0.5$, $\rho = 0.7$, and $\rho = 0.9$.

$\rho = 0.5$		$\mathbb{P}(Q_i = 0)$	$\mathbb{P}(Q_i = 1)$	$\mathbb{P}(Q_i = 2)$	$\mathbb{P}(Q_i = 3)$	$\mathbb{P}(Q_i = 4)$	$\mathbb{P}(Q_i = 5)$	$\mathbb{P}(Q_i = 6)$
$i = 1$	$B = 2$	0.9362	0.0612	0.00249	0.000092	0.000003	0.000000	0.000000
	Sim	0.9362	0.0612	0.00249	0.000093	0.000004	0.000000	0.000000
$i = 2$	$B = 2$	0.8709	0.1179	0.01029	0.00084	0.000071	0.000006	0.000001
	Sim	0.8709	0.1179	0.01028	0.00084	0.000071	0.000006	0.000001
$i = 3$	$B = 2$	0.8055	0.1681	0.0230	0.00295	0.00040	0.00006	0.000008
	Sim	0.8054	0.1681	0.0230	0.00297	0.00040	0.00006	0.000008
$i = 4$	$B = 2$	0.7412	0.2109	0.0395	0.0069	0.00126	0.00024	0.000047
	Sim	0.7411	0.2109	0.0395	0.0069	0.00127	0.00024	0.000049

$\rho = 0.7$		$\mathbb{P}(Q_i = 0)$	$\mathbb{P}(Q_i = 1)$	$\mathbb{P}(Q_i = 2)$	$\mathbb{P}(Q_i = 3)$	$\mathbb{P}(Q_i = 4)$	$\mathbb{P}(Q_i = 5)$	$\mathbb{P}(Q_i = 6)$
$i = 1$	$B = 2$	0.8953	0.0969	0.00722	0.00052	0.000039	0.000003	0.000000
	Sim	0.8952	0.0970	0.00725	0.00052	0.000039	0.000003	0.000000
$i = 2$	$B = 2$	0.7840	0.1798	0.0300	0.00504	0.00090	0.00017	0.000032
	Sim	0.7838	0.1800	0.0301	0.00509	0.00091	0.00017	0.000034
$i = 3$	$B = 2$	0.6724	0.2401	0.0636	0.0171	0.00484	0.00142	0.00043
	Sim	0.6720	0.2400	0.0637	0.0172	0.00493	0.00148	0.00046
$i = 4$	$B = 2$	0.5661	0.2756	0.0994	0.0361	0.0137	0.0054	0.00218
	Sim	0.5655	0.2754	0.0994	0.0362	0.0139	0.0056	0.00228

$\rho = 0.9$		$\mathbb{P}(Q_i = 0)$	$\mathbb{P}(Q_i = 1)$	$\mathbb{P}(Q_i = 2)$	$\mathbb{P}(Q_i = 3)$	$\mathbb{P}(Q_i = 4)$	$\mathbb{P}(Q_i = 5)$	$\mathbb{P}(Q_i = 6)$
	$B = 2$	0.8302	0.1474	0.0195	0.00253	0.00034	0.000047	0.000007
$i = 1$	$B = 3$	0.8298	0.1477	0.0196	0.00255	0.00034	0.000048	0.000007
	Sim	0.8297	0.1477	0.0196	0.00256	0.00035	0.000049	0.000007
	$B = 2$	0.6362	0.2479	0.0779	0.0251	0.0084	0.0029	0.00100
$i = 2$	$B = 3$	0.6350	0.2480	0.0783	0.0254	0.0086	0.0030	0.00105
	Sim	0.6345	0.2481	0.0784	0.0256	0.0087	0.0031	0.00109
	$B = 2$	0.439	0.2669	0.1350	0.0710	0.0387	0.0215	0.0121
$i = 3$	$B = 3$	0.437	0.2654	0.1346	0.0713	0.0393	0.0222	0.0127
	Sim	0.435	0.2644	0.1343	0.0713	0.0397	0.0226	0.0131
	$B = 2$	0.267	0.2159	0.1438	0.0993	0.0712	0.0520	0.0384
$i = 4$	$B = 3$	0.265	0.2129	0.1413	0.0974	0.0701	0.0517	0.0387
	Sim	0.263	0.2109	0.1394	0.0959	0.0690	0.0510	0.0383

Table 6.1: Queue length distributions

	Queue 1	Queue 2	Queue 3	Queue 4
$\rho = 0.5$	0.0000	0.0001	0.0001	0.0002
$\rho = 0.7$	0.0003	0.0005	0.0009	0.0015
$\rho = 0.9$ $(B = 2)$	0.0011	0.0035	0.0138	0.0410
$\rho = 0.9$ $(B = 3)$	0.0002	0.0011	0.0055	0.0191

Table 6.2: Total variation distances

For loads 0.5 and 0.7 the approximation is very accurate even with B as small as 2. The differences in individual probabilities occur only in the 4th decimal or later. For $\rho = 0.9$ and $B = 2$, the approximation is less accurate, but still quite good. If $B = 3$, the approximation again becomes more accurate.

The probabilities obtained from simulation are the average probabilities of 10 simulation runs of $25 \cdot 10^6$ time slots each. Furthermore, each probability in the table is rounded according to the value of the standard deviation σ in the simulation outcomes of that probability. If the first four digits of $\sigma/\sqrt{10}$ are zero, but the fifth is nonzero, then 4 digits are shown, etc.

It will be convenient to express the error of the approximation as a single number for each queue. To this end, we use the total variation distance between the approximated and simulated queue length distribution, which, for queue i, is defined as

$$d_i = \sum_{k=0}^{\infty} |q_{i,k} - \widehat{q}_{i,k}|, \tag{6.5.1}$$

where $q_{i,k}$ and $\widehat{q}_{i,k}$ denote simulation and approximation values for $\mathbb{P}(Q_i = k)$. The values of d_i can be found in Table 6.2.

Because we can approximate the mean waiting times using Little's law, we can also compare our approximation with the existing mean waiting time approximations of Boxma and Meister [34] and Groenendijk and Levy [68]. We do so in Table 6.3. In this table, the approximation of Boxma and Meister is indicated by BM and that of Levy and Groenendijk by LG.

Table 6.3 again illustrates that our approximation is very accurate in general. In all cases, except queue 4 and $\rho = 0.9$, our approximation is more accurate than the Boxma-Meister and Levy-Groenendijk approximations. Both Boxma and Meister [34, Rem. 5.2] and Groenendijk and Levy [68, Sec. IV] give suggestions to improve their approximations for high loads. These suggestions were taken into account in Table 6.3.

Because the state spaces of the SMCs are exponential in N, it is clear that our approximation can only be applied to polling systems with few queues. For our application, networks on chips, however, this does not pose a problem since the switches there typically have only few queues, usually 4 or 5. If $N = 4$ and $B = 2$, the running time of our approximation for one value of ρ and all four queues is only about 2 or 3 seconds. If $B = 3$, the running time increases to roughly 50 seconds. For comparison, the ten simulation runs that give the level of accuracy presented in this section require about 15 to 20 minutes in total, per value of ρ.

$\rho = 0.5$	Queue 1	Queue 2	Queue 3	Queue 4
$B = 2$	0.329	0.413	0.499	0.586
Sim.	0.329	0.413	0.500	0.587
BM	0.346	0.423	0.500	0.577
LG	0.299	0.396	0.498	0.604

$\rho = 0.7$	Queue 1	Queue 2	Queue 3	Queue 4
$B = 2$	0.615	0.854	1.138	1.462
Sim.	0.618	0.858	1.145	1.475
BM	0.709	0.938	1.167	1.395
LG	0.539	0.830	1.152	1.503

$\rho = 0.9$	Queue 1	Queue 2	Queue 3	Queue 4
$B = 2$	1.172	1.98	3.50	6.46
$B = 3$	1.179	2.01	3.59	6.84
Sim.	1.181	2.02	3.66	7.21
BM	1.590	2.71	4.70	5.97
LG	1.168	1.94	3.04	7.71

Table 6.3: Mean waiting times

6.6 Large-scale numerical study

In this section, we perform a numerical experiment on a larger scale to study the accuracy of our approximation. We vary the following six characteristics of the polling system over a number of values: The total load ρ, the number of queues N, the service discipline, the level of symmetry, the arrival processes, and the routing matrix. We consider all possible combinations, i.e., every possible load is combined with every possible value of N, every possible service discipline, and so on. An overview of the values of these characteristics can be found in Table 6.4. In total, the experiment comprises $5 \cdot 4 \cdot 4 \cdot 3 \cdot 3 \cdot 2 = 1440$ polling systems.

We assume that $q(i) = q$, i.e., within one polling system considered in the experiment, all queues have the same service discipline. We further assume that $\rho_i = \nu_i \rho$, where $\sum_i \nu_i = 1$. The constants ν_i are determined by the level of symmetry, which is either symmetric, asymmetric, or very asymmetric: In the symmetric case, every queue gets a fraction $1/N$ of the load; $\nu_i = 1/N$. In the asymmetric

ρ	0.5	0.6	0.7	0.8	0.9
N	2	3	4	5	
q	0	0.3	0.7	1	
Symmetry	Symmetric		Asymmetric	Very asymm.	
Arrival process	Bernoulli		Poisson	Geometric	
Routing	Cyclic		Uniform		

Table 6.4: The numerical experiment.

ρ	0.5	0.6	0.7	0.8	0.9
	0	0	0	0	100
N	2	3	4	5	
	26	27	25	22	
q	0	0.3	0.7	1	
	12	13	21	54	
Symmetry	Symmetric	Asymmetric	Very asymm.		
	59	25	16		
Arrival process	Bernoulli	Poisson	Geometric		
	6	40	54		
Routing	Cyclic	Uniform			
	53	47			

Table 6.5: Systems with the highest errors

case, $(\nu_1, \ldots, \nu_N) \sim (1, 2, \ldots, N)$, where $\sim$ means 'proportional to'. For example, if $N = 2$, $(\nu_1, \nu_2) = (1/3, 2/3)$, if $N = 3$, $(\nu_1, \nu_2, \nu_3) = (1/6, 2/6, 3/6)$, and so on. In the very asymmetric case each queue receives a fraction 0.1 of the load, except queue N which gets the rest.

The distribution of the number of packets arriving to queue i is from the same family for all i, but its mean ρ_i depends on i. The distribution can be Bernoulli with parameter ρ_i, Poisson with parameter ρ_i, or geometric with parameter $1/(1 + \rho_i)$. Here the parameter is $1/(1 + \rho_i)$ so that the mean number of packets arriving each time slot is ρ_i. Note that the geometric distribution we use has positive mass at 0.

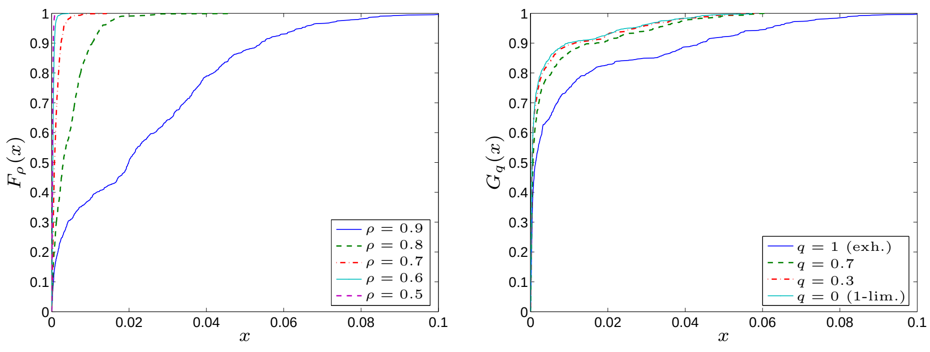
Finally, the routing discipline used is either cyclic or uniform. With uniform routing, if the server leaves a certain queue, it selects one of the other queues at random, each with the same probability, i.e., $P(i, j) = 1/(N - 1)$ for $i \neq j$.

We compare our approximation with $B = 2$ with simulation outcomes. For every queue of a polling system, the error of the approximation is defined as the total variation distance between the approximated and simulated queue length distributions, cf. (6.5.1).

Table 6.5 shows the characteristics of the 100 systems with the largest average error (averaged over the queues). This table should be read as follows: The 100 systems all have $\rho = 0.9$, 26 have two queues, 27 have three, etc.

Table 6.5 reveals that the first and foremost cause of a high error is a high load. Related to this observation, Table 6.5 shows that a higher variance of the arrival process also leads to a higher error. The variances of the Bernoulli, Poisson, and Geometric arrival processes are given by $\rho_i(1 - \rho_i)$, ρ_i, and $\rho_i(1 + \rho_i)$ respectively. The cause of this error is the truncation of queue lengths. As the load or the variance of the arrival processes increases, queue lengths increase as well, and hence the error induced by truncation.

Second, Table 6.5 indicates that the error increases if exhaustive service is used. The pivotal assumption of our approximation is that the contents of the truncated queues go from 'B or more' to $B - 1$ with a fixed probability. With exhaustive service, the time spent serving one queue consecutively is larger than with 1-limited service. As a result, when the server finally starts serving one of the truncated



(a) The function $F_\rho(x)$ **(b)** The function $G_q(x)$

Figure 6.1: The fraction of queues with an error less than x, conditioned on the load ρ and the service discipline parameter q.

queues, this queue will have B or more packets with a larger probability. The error of our approximation is therefore larger if exhaustive service is used.

Third, Table 6.5 suggests that the average error is largest if the system is symmetric. Indeed we found that, on the whole, the average error decreases as the system becomes more asymmetric. Consider, as an extreme example, a polling system where one queue receives almost the entire load, and other queues receive only a very small fraction. In this case, the approximation is indeed very accurate, since the lightly loaded queues hardly ever have 'B or more' packets.

In Fig. 6.1a, we show $F_\rho(x)$, which is defined as the fraction of queues in systems with load ρ, whose error, defined by (6.5.1), is less than x. Fig. 6.1a indeed shows that the most important cause of a high error is a high load.

Likewise, Fig. 6.1b shows the function $G_q(x)$, which is defined as the fraction of queues in systems with service discipline parameter q, whose error is less than x. Fig. 6.1b clearly illustrates that the error is the largest for exhaustive service, whereas there is only a small difference between 1-limited, Bernoulli with parameter 0.3, and Bernoulli with parameter 0.7.

We conclude that for $B = 2$ the error is certainly acceptable for loads up to 0.7 or 0.8, depending on for instance the service discipline and the variance in the arrival process. For higher loads, B should be increased to reduce the error if computationally feasible.

6.7 Implementation with Kronecker products

In this section, we derive alternative expressions for the matrices $A_{i,l}$ and $B_{i,l}$ using Kronecker products. The matrices Ψ_i and $\widetilde{\Psi}_i$ describing the server movements from an empty queue to a non-empty queue are easily determined algorithmically so they will not be dealt with here.

For an $n_A \times m_A$ matrix A and an $n_B \times m_B$ matrix B, the Kronecker product is an $n_A n_B \times m_A m_B$ matrix defined as

$$A \otimes B = \begin{pmatrix} A(1,1)B & A(1,2)B & \dots & A(1,m_A)B \\ \vdots & \vdots & \ddots & \vdots \\ A(n_A,1)B & A(n_A,2)B & \dots & A(n_A,m_A)B \end{pmatrix}.$$

Kronecker products are especially useful in describing Markov chain transitions on multidimensional sets; if transitions on a set $V \times W$ can be decomposed into independent transitions on V and W, then the transition probability matrix on $V \times W$ is $P \otimes Q$, where P and Q are the transition probability matrices on V and W respectively, provided $V \times W$ is ordered lexicographically.

If, more importantly, transitions on W only depend on the current state in V but do not depend on the destination state in V (i.e., a transition from (v,w) to (v',w') occurs with probability $p(v,v')q_v(w,w')$, where $p(v,v')$ is the probability of going from v to v', and $q_v(w,w')$ that of going from w to w' given v), then the transition probability matrix on $V \times W$ is given by

$$\begin{pmatrix} p_1 & \otimes Q_1 \\ p_2 & \otimes Q_2 \\ \vdots & \vdots \\ p_m & \otimes Q_m \end{pmatrix}. \tag{6.7.1}$$

Here, $V = \{1,\dots,m\}$, p_v is a vector with elements $p(v,v')$, and Q_v is a matrix with elements $q_v(w,w')$.

In order to give the alternative expressions for $A_{i,l}$ and $B_{i,l}$, we assume that Φ_i and $\widetilde{\Phi}_i$ are ordered lexicographically. The matrix $A_{i,l}$ describes changes where queue i goes up by $l-1$ levels. This happens if there are either $l-1$ arrivals and no service completion, or l arrivals and a service completion:

$$A_{i,l} = x_i(l-1)D_{i,0} + x_i(l)D_{i,1}, \qquad \text{for } l = 0,1,\dots,$$

where $x_i(-1) := 0$, and $D_{i,0}$ and $D_{i,1}$ are the transition probability matrices within phases, without and with a service completion at queue i, respectively.

The matrix $D_{i,0}$ describes phase transitions *without* a service completion at queue i. Such transitions are caused by changes in the service index, arrivals and a service completion at queue j, and arrivals at the other queues. The changes in the contents of the queues only depend on the service index through the service completion. In particular, the changes in queue contents do not depend on the queue the server moves to in the next time slot, so we obtain, by (6.7.1):

$$D_{i,0} = \begin{pmatrix} r_1 & \otimes Y_{i,1} & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ r_2 & \otimes X_1 & \otimes Y_{i,2} & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ r_{i-1} & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes Y_{i,i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ 0 & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ r_{i+1} & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes Y_{i,i+1} & \otimes \ldots \otimes X_N \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r_N & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes Y_{i,N} \end{pmatrix},$$

where 0 is the length N zero vector.

In the first block row queue 1 is currently in service. The changes in the service index are thus given by r_1, the transitions of queue 1 by $Y_{i,1}$, and the transitions of the other queues by X_k. In the second block row, queue 2 is in service so a $Y_{i,2}$ appears, and so on. The zero vector in the ith block row represents the fact that a transition *without* a service completion at queue i cannot occur if the service index is i.

The matrix $D_{i,1}$ describes phase transitions *with* a service completion at queue i. Using similar arguments as in the derivation of $D_{i,0}$, and observing that a transition *with* a service completion at queue i can only occur if the service index is equal to i, we obtain:

$$D_{i,1} = \begin{pmatrix} 0 & \otimes Y_{i,1} & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ 0 & \otimes X_1 & \otimes Y_{i,2} & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes Y_{i,i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ r_i & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes X_N \\ 0 & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes Y_{i,i+1} & \otimes \ldots \otimes X_N \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \otimes X_1 & \otimes X_2 & \otimes \ldots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \ldots \otimes Y_{i,N} \end{pmatrix}.$$

The matrices $B_{i,l}$ can be expressed as

$$B_{i,l} = x_i(l)\widetilde{D}_i, \qquad \text{for } l = 0, 1, \ldots,$$

where $\widetilde{D}_i$ is the phase transition probability matrix for level $n_i = 0$. Note that, in level $n_i = 0$, there is never a service completion at queue i.

The matrix $\widetilde{D}_i$ is given by:

$$\widetilde{D}_i = \begin{pmatrix}
e_1 & \otimes x_1 & \otimes x_2 & \otimes \dots \otimes x_{i-1} & \otimes x_{i+1} & \otimes \dots \otimes x_N \\
r_1 & \otimes \widetilde{Y}_{i,1} & \otimes X_2 & \otimes \dots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \dots \otimes X_N \\
e_2 & \otimes x_1 & \otimes x_2 & \otimes \dots \otimes x_{i-1} & \otimes x_{i+1} & \otimes \dots \otimes x_N \\
r_2 & \otimes X_1 & \otimes \widetilde{Y}_{i,2} & \otimes \dots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \dots \otimes X_N \\
\vdots & \vdots & & \vdots \quad \vdots & & \vdots \\
e_{i-1} & \otimes x_1 & \otimes x_2 & \otimes \dots \otimes x_{i-1} & \otimes x_{i+1} & \otimes \dots \otimes x_N \\
r_{i-1} & \otimes X_1 & \otimes X_2 & \otimes \dots \otimes \widetilde{Y}_{i,i-1} & \otimes X_{i+1} & \otimes \dots \otimes X_N \\
e_i & \otimes x_1 & \otimes x_2 & \otimes \dots \otimes x_{i-1} & \otimes x_{i+1} & \otimes \dots \otimes x_N \\
e_{i+1} & \otimes x_1 & \otimes x_2 & \otimes \dots \otimes x_{i-1} & \otimes x_{i+1} & \otimes \dots \otimes x_N \\
r_{i+1} & \otimes X_1 & \otimes X_2 & \otimes \dots \otimes X_{i-1} & \otimes \widetilde{Y}_{i,i+1} & \otimes \dots \otimes X_N \\
\vdots & \vdots & & \vdots \quad \vdots & & \vdots \\
e_N & \otimes x_1 & \otimes x_2 & \otimes \dots \otimes x_{i-1} & \otimes x_{i+1} & \otimes \dots \otimes x_N \\
r_N & \otimes X_1 & \otimes X_2 & \otimes \dots \otimes X_{i-1} & \otimes X_{i+1} & \otimes \dots \otimes \widetilde{Y}_{i,N}
\end{pmatrix},$$

where e_j is the jth row of the $N \times N$ identity matrix and the vector x_k is given by $x_k = (x_k(0), \dots, x_k(B-1), \sum_{l \geq B} x_k(l))$.

The first block row (which is a block consisting of one row) represents transitions from state $(1, 0, \dots, 0)$, which is the state where all queues are empty and the server is waiting at queue 1. In this state, the position of the server does not change, i.e., its changes are given by e_1. The changes in the contents of the other queues are given by the independent arrivals to those queues, i.e., by x_k.

The second block row describes transitions from states where queue 1 is non-empty and in service. Like before, changes in the service index are given by r_1, and changes in all queues except queue 1 by X_k. Changes in queue 1 are given by $\widetilde{Y}_{i,1}$ because the process is in level $n_i = 0$. By following similar reasonings for the other block rows, the expression for $\widetilde{D}_i$ follows.

Remark 6.7.1. For the computation of the equilibrium distribution of an SMC of $M/G/1$ type, one has to compute the G-matrix. Several algorithms to do so are available (see, for instance, [22]). In our numerical studies we found that Functional Iterations with the default U-based scheme is the fastest. In particular, this iterative method allows one to start with an initial G-matrix. Starting with the G-matrix as computed in the previous iteration speeds up the computations.

Remark 6.7.2. If the arrival processes are ordinary (i.e., non-batch) Bernoulli arrival processes, we have an SMC of the Quasi-Birth-Death (QBD) type instead of M/G/1 (see, e.g., [91]). In this case, an R-matrix has to be computed, which considerably simplifies actual implementations.

6.8 Conclusion

We devised an algorithm that approximates the marginal queue length distributions in a discrete-time polling system with Bernoulli service and Markovian routing. The key step in this approximation is the translation of the queue length process to a structured Markov chain, where the contents of one queue are stored in the level and truncated contents of the other queues in the phase, i.e., we store whether there are $0, 1, \dots, B-1$, or 'B or more' packets in the other queues. We furthermore use an iterative procedure to determine the probability that the contents of the other queues go from 'B or more' to $B-1$.

As B tends to infinity, the approximation becomes exact. However, with $B = 2$, the approximation is already very accurate in general, as was shown through numerical experiments. Furthermore, the accuracy of the approximation decreases if the load increases, and if the variance in the arrival process increases. In addition to this, the accuracy decreases if $q(i)$ - the probability that after a service completion at queue i, the server serves queue i again - increases. This, for example, implies that the approximation is more accurate for the 1-limited service discipline ($q(i) = 0$) than for the exhaustive service discipline ($q(i) = 1$). In cases where the inaccuracy reaches an unacceptable level, B can be increased further at the cost of a higher running time.

The memory and computation requirements of the approximation are exponential in the number of queues. Clearly, this entails that the approximation is only practical for polling systems with few queues. For networks on chips, however, this does not pose a problem since switches in these networks typically have only few queues.

Throughout this chapter, we assumed that all buffers are infinite. For finite buffers, the same procedure involving queue truncation and iterative determination of the truncation parameters can still be applied. Another interesting extension of the approximation described here is an extension to polling systems with Markovian arrival processes. The extension of our approximation to such systems is straightforward, but the computation time and memory requirements would increase.

Chapter 7

Polling tree networks with flow control

In the previous three chapters, we considered so-called open (networks of) polling stations, where packets arrive from the exterior. In this chapter, we consider a closed network of polling stations, which means that the number of packets in the network from the same source is fixed. Closed queueing networks resemble networks with flow control operating under heavy loads; flow control limits the number of packets from the same source to a *maximum*, and heavy loads imply that served packets are quickly replaced by new packets from the same source, which is modelled by keeping the number of packets from the same source *fixed*.

The network we consider consists of two layers of polling stations. All nodes in the network use the random polling service discipline. With this service discipline, every queue is served with a fixed probability every time slot, independently of what happened in previous time slots. Furthermore, we discuss the extension of our results to the more general case of Bernoulli service and Markovian routing, which contains the cyclic 1-limited service discipline as a special case.

The *total* throughput in our network is always equal to 1 because the last node of the network (the sink) always transmits one packet per time slot. One of the functions of flow control, however, is to achieve fairness [61]. In this chapter we therefore study the *division* of that throughput over packets from various sources, which depends on the flow control limits, buffer sizes, and service disciplines.

7.1 Model

We consider a concentrating tree network consisting of two layers of polling stations with flow control, as displayed in Figure 7.1. Node 0 is the last node of the network, and has N queues. Some of those queues are connected to other nodes and some are not. If queue i of node 0 is connected to another node, then that node is called node i. Node i, $i > 0$, has N_i queues.

Packets in node i and packets in queue i of node 0 are called type i packets. Type i packets are further subdivided into type i,j packets, $j = 1, \dots, N_i$, such that the sub-type of the packet denotes the source of the packets (see Fig. 7.1). If queue i of node 0 is not connected to another node (i.e., if node i does not exist), we define $N_i = 1$.

The network we consider is saturated, which means that as soon as a type i,j packet leaves the network, another type i,j packet immediately arrives. The number of type i,j packets in the network is thus fixed and we denote the number of type i,j packets by $L_{i,j}$. We furthermore define $L_i = (L_{i,1}, \dots, L_{i,N_i})$.

All packets and time slots are of unit size and packet departures and arrivals occur at the end of time slots, with departures before arrivals. The queues of node 0 are finite, and the size of queue i of node 0 is denoted by B_i. If queue i of node 0 is full, packets from node i are blocked. When node 0 transmits a type i packet, a new packet from node i is allowed to enter node 0 at the same time. Queue j of node i is large enough to store all type i,j packets.

All nodes in the network, including node 0, use the 'random polling' service discipline. With this service discipline, every queue has a fixed probability of being served, i.e., every time slot the server of node i serves queue j with probability $P_i(j)$, regardless of what happened in the past, for $i = 0, \dots, N$. If some of the queues of node i are empty, the probability that queue j is served is proportional

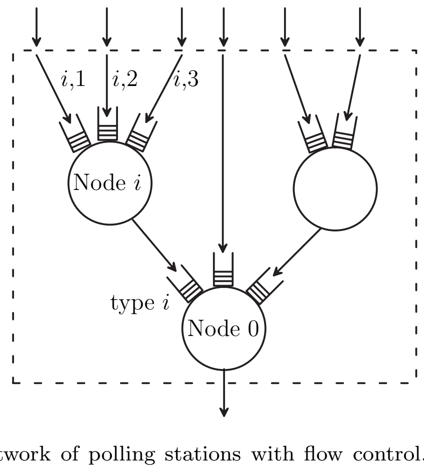


Figure 7.1: A tree network of polling stations with flow control. Due to saturation, the number of packets inside the dashed box is fixed.

to $P_i(j)$, i.e., queue j is served with probability $P_i(j)/\sum_{l\in V_i} P_i(l)$, where V_i is the set of non-empty queues at node i. Within queues, packets are always served in FIFO order. In Section 7.4 we discuss the extension of our work to the more general case with Bernoulli scheduling and Markovian routing, which includes the 1-limited service discipline as a special case.

Queueing networks with blocking have been analysed extensively. For overviews of such models, the reader is referred to Balsamo et al. [11], Dallery and Gershwin [48], Perros [112], and Onvural [110]. Closed queueing networks, and in particular closed queueing networks with blocking, are difficult to analyse exactly. Many different approximations have therefore been devised, such as the maximum entropy method by Kouvatsos and Xenios [87, 88], the exponentialisation method by Yao and Buzacott [150], the variable buffer size method by Suri and Diehl [131], the QNET approximation by Dai and Harrison [45], and approximations by Dallery and Frein [47], and Akyildiz [6].

Pioneering work on closed queueing networks in the context of networks with flow control was done by Reiser [116]. He modelled a computer network operating under window flow control as a closed queueing network and devised an efficient heuristic to compute throughput figures for large networks. Many other studies where networks with flow control are modelled as closed queueing networks have since been performed, see, e.g., Baccelli and Bonald [10], Garetto et al. [60].

As we stated in Section 1.4.2, few attempts have been made to analyse networks of *polling stations*, either closed or open. Altman and Yechiali [7], however, studied a closed *single-station* polling system. In this system, after a packet has been served, it moves from queue i to queue j with a probability that depends on both i and j, regardless of the history. The analysis of Altman and Yechiali was later extended to a polling system with a combination of a fixed population and external arrivals by Armony and Yechiali [8] and to a polling system with breakdowns by Dror and Yechiali [51].

The organisation of this chapter is as follows: In Section 7.2 we describe how the closed queueing network can be modelled as a Markov chain, and we derive its equilibrium distribution. This equilibrium distribution is used to obtain throughput results in Section 7.3. In Section 7.4 we discuss the extension of our results to Bernoulli service and Markovian routing, and we obtain the equilibrium distribution for systems with two queues for the extended model. We perform a numerical analysis of the throughput for systems with random polling in Section 7.5. We draw conclusions in Section 7.6.

7.2 Markov chain analysis

In this section, we describe how node i and node 0 together can be modelled as a Markov chain and we obtain the equilibrium distribution of that Markov chain. This equilibrium distribution is used in Section 7.3 to determine the throughput of different sub-types of packets.

An important observation is that all nodes always try to transmit one packet per time slot if there is one. In particular, this implies that many packets may arrive to node 0 every time slot, but only one packet may leave. In stationarity, there will thus be as many type i packets in queue i of node 0 as possible, restricted by the buffer size B_i and the sum of the flow control limits $\sum_j L_{i,j}$.

In the sequel, we consider only those i for which node i exists; if node i does not exist, all type i packets are type i,1 packets. Moreover, in this case, type i,1 packets are served by node 0 with probability $P_0(i)$ because all queues of node 0 are always non-empty. The throughput of type i,1 packets is hence also equal to $P_0(i)$.

Suppose for a moment that queue i of node 0 is large enough to store at least all type i packets except one, i.e., $B_i \geq \sum_j L_{i,j} - 1$. Because all type i packets (except maybe one) are always in queue i of node, and this queue is served in FIFO order, all type i packets pass through the network in a cyclic manner; once an ordering of packets has been chosen, the packets will always be served in that order. The steady-state behaviour of the network thus depends on the initial configuration of packets and its equilibrium distribution is not unique. Moreover, in this case, the throughput can be calculated straightforwardly, as will be demonstrated in Section 7.3. In the sequel we therefore assume $B_i < \sum_j L_{i,j} - 1$.

If $B_i < \sum_j L_{i,j} - 1$, the state of queue i of node 0 only changes when the server is serving that queue; a new type i packet may only enter queue i of node 0 if another one leaves. For the computation of the throughput, we may thus disregard the times at which the server is at the other queues. We can determine the throughput in a network where node 0 is always serving queue i, and multiply that throughput by the fraction of time node 0 is actually serving queue i.

If node 0 is always serving queue i, node i and node 0 together form a closed tandem network with two nodes and unit service times, as displayed in Figure 7.2. The first node uses the random polling service discipline and has buffers large enough to store all packets, and the second node has a FIFO queue with a buffer of size B_i.

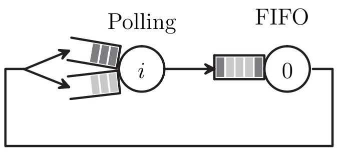


Figure 7.2: The closed tandem network.

We describe the state of this closed tandem network by a finite Markov chain. A state of this Markov chain is a length $B_i + 1$ vector $x = (x_1, \ldots, x_{B_i+1})$. Here, x_k, for $k = 1, \ldots, B_i$, describes the type of the packet at position k of the buffer of node 0, i.e., $x_1 = j$ if the first packet is a type i,j packet, and so on. The last element x_{B_i+1} describes the type of the packet that is going to be served by node i and will enter node 0 in the next time slot.

It is easily verified that this process is indeed Markovian: Node 0 always serves the first packet in the queue and the probability that node i serves queue j depends only on which queues are empty, which can be derived from x using the fact that there are $L_{i,j}$ type i,j packets in the network in total.

The state space of the Markov chain is given by:

$$\Omega_i = \left\{x \in \{1, \dots, N_i\}^{B_i+1} : k_j(x) \le L_{i,j}\right\}, \tag{7.2.1}$$

where $k_j(x)$ is the number of j's in x. The condition $k_j(x) \le L_{i,j}$ reflects that there are at most $L_{i,j}$ packets in the buffer of node 0, and, moreover, if there are indeed $L_{i,j}$ packets in that buffer, the next packet served by node i (i.e., element x_{B_i+1}) must be of a different type.

We can now determine the equilibrium distribution of this Markov chain, denoted by $\pi_i(\cdot)$:

Lemma 7.2.1. *Suppose $B_i < \sum_j L_{i,j} - 1$. Recall that $k_l(x)$ is the number of l's in x. Then, for all $x \in \Omega_i$,*

$$\pi_i(x) = \frac{1}{C_i(B_i, L_i)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)}, \tag{7.2.2}$$

where $C_i(B_i, L_i)$ is the normalisation constant, given by

$$C_i(B_i, L_i) = \sum_{x \in \Omega_i} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)}. \tag{7.2.3}$$

Proof. The equilibrium distribution is given by the unique normalised solution to the balance equations of the Markov chain:

$$\pi_i(x) = \sum_{j \in V_i(x)} \pi_i(j, x_1, \dots, x_{B_i}) \frac{P_i(x_{B_i+1})}{\sum\limits_{k \in V_i(x)} P_i(k)}, \qquad x \in \Omega_i, \tag{7.2.4}$$

where $V_i(x)$ is the set of non-empty queues of node i in state x. The balance equations can be obtained by observing that every transition of the Markov chain consists of shifting packets at node 0 one buffer position ahead and by choosing the queue that will be served by node i in the next time slot. In other words, in state $(j, x_1, \dots, x_{B_i})$, the vector $(x_1, \dots, x_{B_i})$ is shifted towards positions 1 through B_i of the buffer. The probability that a type i, x_{B_i+1} packet is served at node i is given by the rightmost factor of (7.2.4).

Substituting Equation (7.2.2) in the right hand side of (7.2.4) yields

$$\begin{aligned}
\pi_i(x) &= \frac{1}{C_i(B_i, L_i)} \sum_{j \in V_i(x)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)} \frac{P_i(j)}{P_i(x_{B_i+1})} \frac{P_i(x_{B_i+1})}{\sum\limits_{k \in V_i(x)} P_i(k)} \\
&= \frac{1}{C_i(B_i, L_i)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)} \sum_{j \in V_i(x)} \frac{P_i(j)}{\sum\limits_{k \in V_i(x)} P_i(k)} \\
&= \frac{1}{C_i(B_i, L_i)} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)},
\end{aligned}$$

which completes the proof. □

Example 7.2.2. Consider the situation with $B_i = 2$, and $L_i = (2, 2)$, and write $P_i(1) = p_1$ and $P_i(2) = p_2$. In this case, the state space Ω_i consists of the following six states: $(1, 1, 2)$, $(1, 2, 1)$, $(2, 1, 1)$, $(2, 2, 1)$, $(2, 1, 2)$, and $(1, 2, 2)$. In state $(1, 1, 2)$, two type i,1 packets occupy the two buffer positions and a type i,2 packet will move to the last buffer position in the next time slot. The first three states all have probability $p_1^2 p_2 / C_i(B_i, L_i)$, and the last three all have probability $p_1 p_2^2 / C_i(B_i, L_i)$. Because the sum of all probabilities is 1, it follows that $C_i(B_i, L_i) = 3p_1^2 p_2 + 3p_1 p_2^2 = 3p_1 p_2$.

Remark 7.2.3. The network studied in this section can also be seen as a variant of a classical urn problem: Consider N_i urns such that urn j contains $L_{i,j}$ balls, for $j = 1, \dots, N_i$. Suppose that $B_i + 1$ balls are drawn from different urns, and a ball from urn j is drawn with probability $P_i(j)/\sum_{l \in V_i} P_i(l)$ where V_i is the set of non-empty urns. After $B_i + 1$ balls have been drawn, the first ball is placed back in its original urn and a new ball is drawn. Then the second ball is placed back in its urn and a new ball is drawn, and so on. The equilibrium probability that, at an arbitrary point in time, the $B_i + 1$ most recently drawn balls originate from urns $x_1, \dots, x_{B_i+1}$ is given by $\pi_i(x)$.

The normalisation constant $C_i(B_i, L_i)$ can be computed efficiently using the following recursion:

$$C_i(b, (L_{i,1}, \dots, L_{i,n})) = \sum_{k=0}^{L_{i,n}} \binom{b+1}{k} (P_i(n))^k C_i(b-k, (L_{i,1}, \dots, L_{i,n-1})), \quad (7.2.5)$$

for any nonnegative integer $b \le B_i$, and any positive integer $n \le N_i$, with $\binom{b+1}{k} := 0$ for $k > b + 1$.

This recursion can be obtained by conditioning on k, the number of type i,n packets in a system with buffer size b and flow control limits $(L_{i,1}, \dots, L_{i,n})$. In that system, there are $\binom{b+1}{k}$ states with k type i,n packets, each of which induces a factor $(P_i(n))^k$. Furthermore, given that there are k type i,n packets in a certain state x, the remaining $b + 1 - k$ vector positions of x are filled with remaining packet types, restricted by their flow control limits. This leads to a factor $C_i(b - k, (L_{i,1}, \dots, L_{i,n-1}))$, where

$$C_i(-1, (L_{i,1}, \dots, L_{i,n-1})) := 1,$$

i.e., if $k = b + 1$, there are no remaining vector positions to be filled, and a factor 1 is required.

It might happen that $b - k \ge L_{i,1} + \dots + L_{i,n-1}$. In this case, the $b + 1 - k$ remaining vector positions have to be filled with at most $L_{i,1} + \dots + L_{i,n-1} < b + 1 - k$ packets. In other words, there are more positions in the vector than elements to fill the vector with, so we define:

$$C_i(b, (L_{i,1}, \dots, L_{i,n})) = 0, \qquad \text{for } b \ge L_{i,1} + \dots + L_{i,n}, \text{ and all } n.$$

Finally, the behaviour on the boundary $n = 1$ follows from the definition of $C_i(\cdot,\cdot)$ (Eq. (7.2.3)):

$$C_i(b, (L_{i,1})) = (P_i(1))^{b+1}, \qquad \text{for } b < L_{i,1}.$$

7.3 Throughput computation

In this section, we determine the throughput of type i,j packets. As we argued in Section 7.2, this throughput is equal to the throughput in a network where node 0 always serves queue i, multiplied by the fraction of time node 0 is actually serving queue i. Because all queues of node 0 are always non-empty, node 0 serves queue i with probability $P_0(i)$, so we obtain:

$$\gamma_{i,j} = P_0(i)\widetilde{\gamma}_{i,j}, \tag{7.3.1}$$

where $\widetilde{\gamma}_{i,j}$ is the throughput of type i,j packets in a network where node 0 is always serving queue i.

It thus remains to determine $\widetilde{\gamma}_{i,j}$. First, we argued in Section 7.2 that if $B_i \geq \sum_j L_{i,j} - 1$, all packets move cyclically through the network without any change in their ordering. This implies that the fraction of time a type i,j packet is served is equal to the fraction of type i packets that are of type i,j, i.e., $\widetilde{\gamma}_{i,j} = L_{i,j}/\sum_k L_{i,k}$. Hence:

Proposition 7.3.1. *If $B_i \geq \sum_j L_{i,j} - 1$,*

$$\gamma_{i,j} = P_0(i)\frac{L_{i,j}}{\sum\limits_k L_{i,k}}. \tag{7.3.2}$$

If $B_i < \sum_j L_{i,j} - 1$, the throughput of type i,j packets can be determined using the results of Section 7.2:

Proposition 7.3.2. *If $B_i < \sum_j L_{i,j} - 1$,*

$$\gamma_{i,j} = P_0(i)P_i(j)\frac{C_i(B_i - 1, L_i - e_j)}{C_i(B_i, L_i)}, \tag{7.3.3}$$

where e_j is a vector of which all elements are zero, except the jth, which is 1.

Proof. The throughput of type i,j packets is given by the probability that a type i,j packet is served by node 0:

$$\widetilde{\gamma}_{i,j} = \sum_{\substack{x\in\Omega_i\\x_1=j}} \pi_i(x) = \frac{1}{C_i(B_i, L_i)} \sum_{\substack{x\in\Omega_i\\x_1=j}} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x)}. \tag{7.3.4}$$

For all x over which the sum is taken, the first element is a j. The remaining elements are restricted by the flow control limits, but are otherwise chosen arbitrarily. We thus obtain:

$$\begin{aligned}\widetilde{\gamma}_{i,j} &= \frac{1}{C_i(B_i, L_i)} P_i(j) \sum_{\substack{x \in \Omega_i \\ x_1 = j}} \prod_{l=1}^{N_i} (P_i(l))^{k_l(x_2,\ldots,x_{B_i+1})} \\ &= \frac{C_i(B_i - 1, L_i - e_j)}{C_i(B_i, L_i)} P_i(j).\end{aligned}$$

Equation (7.3.3) follows by combining the above expression with (7.3.1). □

In case the buffer sizes are small enough, we can reduce Equation (7.3.3) to a much simpler expression:

Proposition 7.3.3. *If $B_i < L_{i,j}$ for all j,*

$$\gamma_{i,j} = P_0(i) P_i(j). \tag{7.3.5}$$

Proof. If $B_i < L_{i,j}$ for all j, the number of j's in $(x_1, \ldots, x_{B_i+1})$ can never exceed $L_{i,j}$. The restriction that $k_j(x) \leq L_{i,j}$ can thus be removed from the definition of Ω_i, the state space of the Markov chain. It then follows from Newton's binomium that $C_i(B_i, L_i) = (\sum_j P_i(j))^{B_i+1} = 1$, and likewise $C_i(B_i - 1, L_i - e_i) = 1$. Equation (7.3.3) then yields $\gamma_{i,j} = P_0(i)P_i(j)$.

A more intuitive explanation is the following: The throughput of type i,j packets, $\widetilde{\gamma}_{i,j}$, is also given by the probability that node i serves queue j. If $B_i < L_{i,j}$ for all j, queue i of node 0 is too small to store all type i,j packets, for any j, so all queues of node i are always non-empty. Queue j is thus served by node i with probability $P_i(j)$. □

Using these throughput results, the mean total sojourn time and the number of packets in queue i of node 0 can be found easily using Little's law. As there are $L_{i,j}$ type i,j packets in the network, the total mean sojourn time is equal to

$$\mathbb{E}[T_{i,j}] = \frac{L_{i,j}}{\gamma_{i,j}}.$$

Furthermore, every type i packet spends on average $B_i/P_0(i)$ time slots in queue i of node 0, so the mean number of type i,j packets in that queue is given by

$$\mathbb{E}[Q^{(0)}_{i,j}] = \gamma_{i,j} \frac{B_i}{P_0(i)} = B_i \widetilde{\gamma}_{i,j}. \tag{7.3.6}$$

7.4 Bernoulli service and Markovian routing

In this section, we discuss the extension of the results from the previous sections to a network of polling systems with Bernoulli service and Markovian routing. We assume that, after service of queue j, the server of node i, $i = 0, \ldots, N$, serves queue j again with probability $q_i(j)$, and moves to another queue with probability $1-q_i(j)$. If the server moves to another queue, it moves to queue k with probability $P_i(j,k)$, where $P_i(j,j)=0$. If queue k is empty, the server keeps moving according to matrix P_i until it finds a non-empty queue. These movements happen instantaneously. To prevent a situation where some queues cannot be reached by the server, we assume P_i is irreducible for all i.

We introduce the matrix R_i, defined as follows:

$$R_i(j,k) = \begin{cases} q_i(j), & \text{if } j = k, \\ (1-q_i(j))P_i(j,k), & \text{if } j \neq k. \end{cases}$$

Conditioned on the event that all queues of node i are always non-empty, the position of the server of node i constitutes a Markov chain on $\{1,\dots,N_i\}$, with transition probability matrix R_i. Under this condition, the probability that node i is serving queue j at an arbitrary point in time is thus given by $\sigma_i(j)$, where $\sigma_i(j)$ is the unique normalised solution to the following set of equations:

$$\sigma_i(j) = \sum_k \sigma_i(k) R_i(k,j), \tag{7.4.1}$$

for $i = 0,\dots,N$.

Like in the case of random polling, we may disregard the times at which the server of node 0 is not serving queue i for the computation of the throughput. We may thus again consider a closed tandem network of two nodes with unit service times (see Figure 7.2). The first node, however, is now a node with a Bernoulli service discipline and Markovian routing, and the second node still consists of a FIFO queue with a finite buffer.

The state space and state description of the Markov chain describing this network remain the same, i.e., we consider states $x = (x_1,\dots,x_{B_i+1}) \in \Omega_i$. In case node i has two queues, we can determine the equilibrium distribution of this Markov chain. Although we cannot determine the equilibrium distribution analytically for systems with more than 2 queues, the equilibrium distribution can still be obtained numerically for such systems, provided the buffers and the number of queues are small.

Lemma 7.4.1. *Suppose that $B_i < \sum_j L_{i,j} - 1$ and that node i only has two queues. Then,*

$$\pi_i(x) = \begin{cases} C_i \dfrac{\sigma_i(1)}{R_i(2,1)} R_i(k,x_1) \displaystyle\prod_{j=1}^{B_i} R_i(x_j,x_{j+1}), & \text{if } V_i(x) = \{k\}, \text{ for some } k, \quad (7.4.2\text{a}) \\ C_i \sigma_i(x_1) \displaystyle\prod_{j=1}^{B_i} R_i(x_j,x_{j+1}), & \text{if } V_i(x) = \{1,2\}, \quad (7.4.2\text{b}) \end{cases}$$

where C_i is the normalisation constant. Recall that $V_i(x)$ is the set of non-empty queues of node i in state x and note that not all queues can be empty at the same time (i.e., $V_i(x) = \emptyset$) due to the assumption that $B_i < L_{i,1} + L_{i,2} - 1$.

Proof. We prove the lemma by substituting Equations (7.4.2a) and (7.4.2b) in the balance equations of the Markov chain. These balance equations are given by

$$\pi_i(x) = \begin{cases} \pi_i(j, x_1, \dots, x_{B_i}) & \text{if } V_i(x) = \{j\}, \text{ for some } j, \\ \sum_{j=1}^{2} \pi_i(j, x_1, \dots, x_{B_i}) R_i(x_{B_i}, x_{B_i+1}) & \text{if } V_i(x) = \{1, 2\}. \end{cases} \tag{7.4.3}$$

This can be argued as follows: If only one queue of node i, say queue j, is non-empty in state x, a type i,j packet must have been served by node 0 in the previous time slot. After all, if a packet of another type would have been served by node 0, that packet would arrive at node i in the next time slot, and its queue could not be empty. This explains the first line of (7.4.3). If two queues are non-empty, the server moves from queue x_{B_i} to queue x_{B_i+1} with probability $R_i(x_{B_i}, x_{B_i+1})$. This explains the second line of (7.4.3).

In order to determine which one of (7.4.2a) and (7.4.2b) we have to substitute on the right hand side of the balance equations, we need to determine $V_i(j, x_1, \dots, x_{B_i})$. We have:

$$V_i(j, x_1, \dots, x_{B_i}) = \begin{cases} V_i(x) \cup \{x_{B_i}\} \setminus \{j\}, & \text{if } j \neq x_{B_i} \text{ and } j \in W_i(x), \\ V_i(x) \cup \{x_{B_i}\}, & \text{otherwise}, \end{cases} \tag{7.4.4}$$

where $W_i(x) \subseteq V_i(x)$ is, for state x, the set of queues of node i with precisely one packet. We derive this equation for the case where node i has an arbitrary number of queues: Suppose that the system is in state x at time t and in state $(j, x_1, \dots, x_{B_i})$ at time $t-1$. Because at time t there is only an arrival at queue j and a service completion at queue x_{B_i}, all other queues are empty at time t if and only if they are empty at time $t-1$. Queue x_{B_i} is served at $t-1$ so it must be non-empty at time $t-1$. Queue j of node i sees the arrival of a new packet at time t, so it is empty at $t-1$ and non-empty at t if and only if it has precisely one packet at t (i.e., $j \in W_i(x)$) and was not served by node 0 at $t-1$ ($j \neq x_{B_i}$).

We now assume $V_i(x) = \{k\}$, and derive Equation (7.4.2a). Without loss of generality, we furthermore assume that $k = 1$. It follows from the balance equations (7.4.3) that

$$\pi_i(x) = \pi_i(1, x_1, \dots, x_{B_i}).$$

Because queue 2 is empty ($V_i(x) = \{1\}$), queue 1 has more than one packet in it (so $W_i(x) = \emptyset$). After all, if queue 1 has one packet in it and queue 2 zero, $B_i = L_{i,1} + L_{i,2} - 1$, which was excluded by assumption. From Equation (7.4.4) it follows that $V_i(1, x_1, \dots, x_{B_i}) = \{1\}$ if $x_{B_i} = 1$ and $\{1, 2\}$ if $x_{B_i} = 2$. By substituting Equations (7.4.2a) and (7.4.2b) respectively on the right hand side of the balance equations, we obtain:

$$\pi_i(x) \begin{cases} C_i \frac{\sigma_i(1)}{R_i(2,1)} R_i(1,1) R_i(1, x_1) \prod_{j=1}^{B_i-1} R_i(x_j, x_{j+1}) & \text{if } x_{B_i} = 1, \\ C_i \sigma_i(1) R_i(1, x_1) \prod_{j=1}^{B_i-1} R_i(x_j, x_{j+1}) & \text{if } x_{B_i} = 2. \end{cases}$$

Finally, observe that $x_{B_i+1} = 1$ because $V_i(x) = \{1\}$ (i.e., if queue 1 is the only non-empty queue, it has to be served), so the two expressions above indeed correspond to (7.4.2a).

We assume now that $V_i(x) = \{1, 2\}$ and derive Equation (7.4.2b). For $V_i(x) = \{1, 2\}$, the balance equations are given by:

$$\pi_i(x) = \sum_{j=1}^{2} \pi_i(j, x_1, \ldots, x_{B_i}) R_i(x_{B_i}, x_{B_i+1}).$$

Using (7.4.4), we can determine for each j and x_{B_i} whether we have to substitute (7.4.2a) or (7.4.2b) on the right hand side of the expression above. We obtain:

$$\pi_i(x) = \begin{cases} C_i \left[\frac{\sigma_i(1)}{R_i(2,1)} R_i(2,1) R_i(1, x_1) + \sigma_i(2) R_i(2, x_1)\right] \prod_{j=1}^{B_i} R_i(x_j, x_{j+1}), \\ \qquad\qquad\qquad\qquad \text{if } x_{B_i} = 2 \text{ and } 1 \in W_i(x), \\ C_i \left[\sigma_i(1) R_i(1, x_1) + \frac{\sigma_i(1)}{R_i(2,1)} R_i(1,2) R_i(2, x_1)\right] \prod_{j=1}^{B_i} R_i(x_j, x_{j+1}), \\ \qquad\qquad\qquad\qquad \text{if } x_{B_i} = 1 \text{ and } 2 \in W_i(x), \\ C_i \left[\sigma_i(1) R_i(1, x_1) + \sigma_i(2) R_i(2, x_1)\right] \prod_{j=1}^{B_i} R_i(x_j, x_{j+1}), \\ \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases}$$

From Equation (7.4.1), we get $\frac{\sigma_i(1)}{R_i(2,1)} R_i(1,2) = \sigma_i(2)$, which, for all three cases, leads to

$$\begin{aligned} \pi_i(x) &= C_i \left[\sigma_i(1) R_i(1, x_1) + \sigma_i(2) R_i(2, x_1)\right] \prod_{j=1}^{B_i} R_i(x_j, x_{j+1}) \\ &= C_i \sigma_i(x_1) \prod_{j=1}^{B_i} R_i(x_j, x_{j+1}), \end{aligned}$$

again by Equation (7.4.1). □

Remark 7.4.2. Random polling is a special case of Bernoulli service with Markovian routing, where $R_i(j,k) = P_i(k)$ and hence $\sigma_i(k) = P_i(k)$. It is easily verified that Equation (7.2.2) is indeed a special case of Equations (7.4.2a) and (7.4.2b).

The throughput results of Section 7.3 can be straightforwardly extended to systems with Bernoulli service and Markovian routing. As with random polling, the packets move cyclically through the network if the buffer of node i is large enough to store all type i packets except 1. Because node 0 is always non-empty, the probability that node 0 is serving queue i is $\sigma_0(i)$, and we obtain:

Proposition 7.4.3. *If $B_i \geq \sum_j L_{i,j} - 1$,*

$$\gamma_{i,j} = \sigma_0(i) \frac{L_{i,j}}{\sum_k L_{i,k}}. \tag{7.4.5}$$

If the buffers are not large enough, the throughput is obtained using the equilibrium distribution of the Markov chain, which, although not analytically available, can be determined numerically for small systems. To be precise, we have:

Proposition 7.4.4. *If $B_i < \sum_j L_{i,j} - 1$,*

$$\gamma_{i,j} = \sigma_0(i) \sum_{\substack{x \in \Omega_i \\ x_1 = j}} \pi_i(x). \tag{7.4.6}$$

Finally, if the buffer is small a further simplification is possible, similar to Proposition 7.3.3 for random polling. Namely, if the buffer is too small to store all type i,j packets, for any j, all queues of node i are always non-empty so the probability that a type i,j packet is served by node i (and hence the probability that $x_1 = j$) is $\sigma_i(j)$. We thus obtain:

Proposition 7.4.5. *If $B_i < L_{i,j}$ for all j,*

$$\gamma_{i,j} = \sigma_0(i)\sigma_i(j). \tag{7.4.7}$$

7.5 Numerical analysis

The analysis of the previous sections illustrates that the division of throughput is determined by an interaction between the flow control mechanism and the service disciplines: For small buffers, the throughput division is determined by the service disciplines, for large buffers by the flow control mechanism, and for intermediate buffers by a mix of the two. For intermediate buffers the throughput division can be determined using the Markov chain approach. This approach by itself, however, offers little insight in the precise interaction between the flow control limitations and the service disciplines. In this section, we therefore study how the throughput depends on the different parameters through a numerical study.

We consider a network in which node i has four queues and uses random polling. We focus on the throughput given that node 0 is always serving queue i, i.e., on $\widetilde{\gamma}_{i,j}$. The actual throughput in the network can easily be found by multiplying that throughput by the probability that node 0 is serving queue i. The running example of this section has the following parameters: $B_i = 32$, $P_i = (0.1, 0.2, 0.3, 0.4)$, and $L_i = (20, 16, 12, 8)$. We vary one of these parameters at a time, and study how this affects the throughput division.

First, we show the throughput division in Figure 7.3 for B_i ranging from 1 to 64. This figure clearly illustrates that the throughput is determined by the service

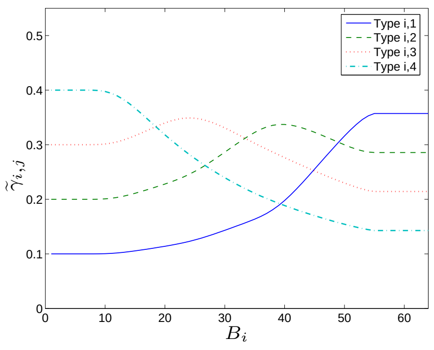


Figure 7.3: The division of throughput for various buffer sizes.

disciplines for small buffers, and by the flow control mechanism for large buffers. However, the transition from the small buffer regime to the large buffer regime does not occur monotonously; the throughput of some packet types first increases and later decreases.

We can explain this phenomenon by looking at the number of packets in queue i of node 0. As the buffer size grows, the number of type i,j packets in queue i of node 0 gets closer to the maximal number allowed by flow control. The flow control restrictions thus cause some queues to remain empty and the throughput of corresponding packet types to decrease. If some queues are empty, the other non-empty queues are served with a higher probability and the throughput of the corresponding packet types will increase. As the buffer size grows even further, the flow control restrictions on these packet types will become stronger as well, which can cause their throughput to fall again.

For example, if B_i is equal to twelve, the mean number of type i,4 packets in queue i of node 0 is roughly five (see Equation (7.3.6)), out of eight allowed by flow control. On the other hand, the mean number of type i,3 packets in queue i of node 0 is roughly four, out of twelve allowed. The flow control restrictions on type i,4 packets are thus more severe than those on type i,3 packets. As the buffer size B_i increases further, the throughput of type i,4 will hence decrease and that of type i,3 will grow.

When B_i is twenty-five, there are on average roughly eight type i,3 packets in queue i of node 0, out of twelve allowed. Flow control restrictions have thus become more severe for this type, and its throughput will decrease again.

That some packet types benefit from the flow control restrictions on other packet types is especially clear for type i,1 packets. Queue 1 of node i is served with a 0.1 probability, and only when the other queues are frequently empty due to the flow control restrictions, type i,1 packets will receive a more significant part of the throughput. Figure 7.3 clearly illustrates this: The throughput of type i,1 experiences a sharp increase when the throughputs of all other types decrease.

We consider again the running example, i.e., we set $B_i = 32$. In Table 7.1,

	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$P_i(j)$	0.1	0.2	0.3	0.4
$L_{i,j}$	20	16	12	8
$\widetilde{\gamma}_{i,j}$	0.1512	0.3016	0.3198	0.2274
$\mathbb{E}[Q^{(0)}_{i,j}]$	4.84	9.65	10.23	7.28
$\mathbb{E}[Q^{(0)}_{i,j}]/L_{i,j}$	0.24	0.60	0.85	0.91

Table 7.1: Parameters and performance measures of the running example.

we show $\mathbb{E}[Q^{(0)}_{i,j}]$, the expected number of type i,j packets in node 0, as well as $\mathbb{E}[Q^{(0)}_{i,j}]/L_{i,j}$, the expected number of type i,j packets in node 0 relative to its maximum. Based on the discussion above, the latter can be seen as a (rough) measure for how strong the flow control restrictions are; the closer to 1, the stronger the flow control restrictions. Table 7.1 reveals that type i,1 packets are not strongly restricted by flow control, but type i,4 packets are; the expected number of packets in node 0 relative to its maximum is 0.24 and 0.91 for type i,1 and type i,4 respectively.

We proceed by investigating the sensitivity of the throughput of these packet types on changes in the flow control limits and service disciplines for the running example, i.e., we vary one of the values of $L_{i,j}$ and $P_i(j)$ for $j = 1, 4$. Here, the value of $P_i(j)$ is varied for one specific j, while maintaining the *ratios* between the other values of $P_i(k)$, for $k \neq j$, and maintaining a total sum of 1.

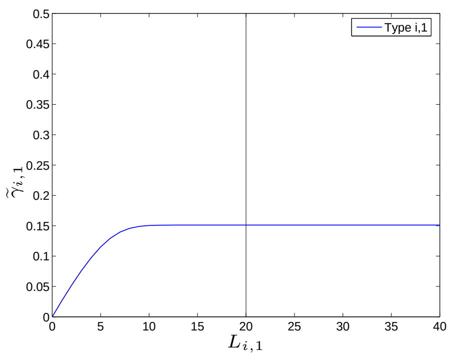
Figures 7.4a and 7.4b show the throughput of type i,1 packets as a function of $P_i(1)$ and $L_{i,1}$, respectively. For both figures, a vertical line marks the value on the x-axis corresponding to the running example. Clearly, the throughput of type i,1 packets is not sensitive to a change in its flow control limit but it is sensitive to a change in the service discipline; the throughput hardly changes if the flow control limit is changed but it does change if the service discipline is changed.

In contrast, Figures 7.5a and 7.5b show the sensitivity of type i,4 throughput on changes in its flow control limit and the service discipline, where the running example is again marked by a vertical line. The throughput of type i,4 is not sensitive to a change in the service discipline, but it is sensitive to a change in the flow control limit.

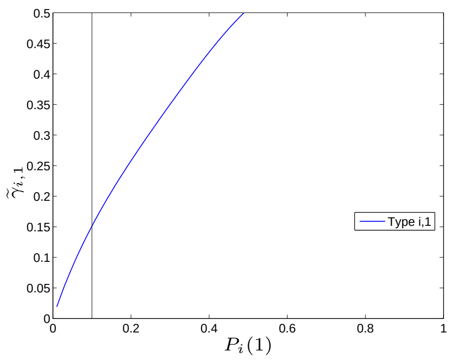
These observations can be used to decide whether flow control limits or service disciplines have to be adjusted when the throughput division over various packet types has to be changed. If a packet type is strongly restricted by the flow control mechanism, the flow control limits should be changed, and otherwise the service discipline of node i should be changed.

7.6 Conclusion

We modelled a saturated concentrating tree network with flow control as a closed network. The equilibrium distribution of the Markov chain describing this network can be derived in closed form for the random polling discipline and for polling

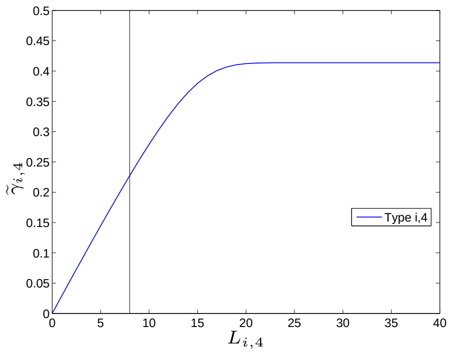


(a) Throughput as a function of $L_{i,1}$.

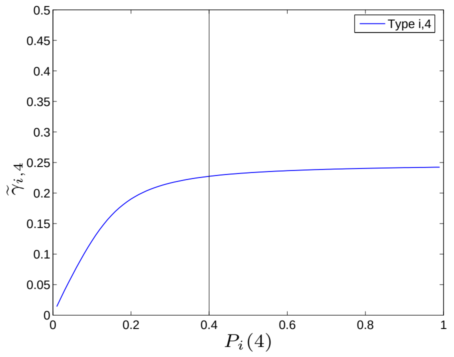


(b) Throughput as a function of $P_i(1)$.

Figure 7.4: The type i,1 throughput as a function of $L_{i,1}$ and $P_i(1)$. The vertical line corresponds to the 'running example'.



(a) Throughput as a function of $L_{i,4}$.



(b) Throughput as a function of $P_i(4)$.

Figure 7.5: The type i,4 throughput as a function of $L_{i,4}$ and $P_i(4)$. The vertical line corresponds to the 'running example'.

stations with two queues and Bernoulli service combined with Markovian routing. Although we could not determine the equilibrium distribution analytically for polling stations with more than two queues and Bernoulli service combined with Markovian routing, the latter can be obtained numerically for small systems.

The division of throughput over various packet types can be determined using the results of the Markov chain approach. For small buffers, this throughput division is fully determined by the service disciplines, and for large buffers by the flow control mechanism. For intermediate buffers the throughput division is steered by an interaction between the two. We performed a numerical study that provides more insight in the specifics of this interaction.

The numerical study showed that the transition from the small buffer regime to the large buffer regime does not necessarily occur monotonously. If there are many packets of the same type in node 0, flow control restrictions become stronger, which limits the throughput of that type. The throughput that becomes available as a result is divided over the other packet types for which the flow control restrictions are less severe, which causes their throughput to rise. If the buffer size is increased even further, the number of packets of those types in node 0 also increases, which causes flow control limitations to become stronger and their throughput to decrease again.

The expected number of packets of different types in node 0 provides insight in the extent to which packet types are restricted by flow control, which in turn determines which parameters should be changed if the throughput division has to be adjusted: If almost all packets of the same type are in node 0, flow control plays a key role. Changing flow control limits will thus affect the throughput of that type, but changing the service discipline parameters will not. On the other hand, if not many packets are in node 0, service disciplines play a key role. Changing flow control limits thus has little effect, but changing service discipline parameters has a large effect.

The phenomena described above were only analysed for the network with two layers of polling stations and random polling. These phenomena, however, occur in more complex tree networks as well; additional simulations have revealed that they also occur in networks of three layers and networks with the cyclic 1-limited service discipline.

References

[1] I.J.B.F. Adan, *A Compensation Approach for Queueing Problems*, Ph.D. thesis, Eindhoven University of Technology, 1991.

[2] I.J.B.F. Adan, O.J. Boxma, and J.A.C. Resing, *Queueing models with multiple waiting lines*, Queueing Systems **37** (2001), no. 1–3, pp. 65–98.

[3] I.J.B.F. Adan and J.A.C. Resing, *Queueing Theory*, Lecture notes, Eindhoven University of Technology, `www.win.tue.nl/~iadan/queueing.pdf`.

[4] I.J.B.F. Adan, J. Wessels, and W.H.M. Zijm, *A compensation approach for two-dimensional Markov processes*, Advances in Applied Probability **25** (1993), no. 4, pp. 783–817.

[5] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C.A. Zeferino, *SPIN: A scalable, packet switched, on-chip micro-network*, Proc. of DATE, 2003, pp. 70–73.

[6] I.F. Akyildiz, *On the exact and approximate throughput analysis of closed queuing networks with blocking*, IEEE Transactions on Software Engineering **14** (1988), no. 1, pp. 62–70.

[7] E. Altman and U. Yechiali, *Polling in a closed network*, Probability in the Engineering and Informational Sciences **8** (1994), pp. 327–343.

[8] R. Armony and U. Yechiali, *Polling systems with permanent and transient jobs*, Stochastic Models **15** (1999), no. 3, pp. 395–427.

[9] B. Avi-Itzhak, *A sequence of service stations with arbitrary input and regular service times*, Management Science **11** (1965), no. 5, pp. 565–571.

[10] F. Baccelli and T. Bonald, *Window flow control in FIFO networks with cross traffic*, Queueing Systems **32** (1999), no. 1, pp. 195–231.

[11] S. Balsamo, V. De Nitto Personé, and R.O. Onvural, *Analysis of Queueing Networks with Blocking*, Kluwer Academic Publishers, Dordrecht, 2001.

[12] P. Beekhuizen, *Mathematical modelling of networks-on-chips with best effort traffic*, Master's thesis, Eindhoven University of Technology, November 2005.

[13] P. Beekhuizen, T.J.J. Denteneer, and I.J.B.F. Adan, *Analysis of a tandem network model of a single-router network-on-chip*, Annals of Operations Research **162** (2008), no. 1, pp. 19–34.

[14] P. Beekhuizen, T.J.J. Denteneer, and J.A.C. Resing, *End-to-end delays in polling tree networks*, Proc. of ValueTools, 2008.

[15] P. Beekhuizen, T.J.J. Denteneer, and J.A.C. Resing, *Reduction of a polling network to a single node*, Queueing Systems **58** (2008), no. 4, pp. 303–319.

[16] P. Beekhuizen and J.A.C. Resing, *A saturated tree network of polling stations with flow control*, Submitted for publication.

[17] P. Beekhuizen and J.A.C. Resing, *Approximation of discrete-time polling systems via structured Markov chains*, Proc. of SMCTools, 2009.

[18] P. Beekhuizen and J.A.C. Resing, *Performance analysis of small non-uniform packet switches*, Performance Evaluation **66** (2009), no. 11, pp. 640–659.

[19] L. Benini and G. de Micheli, *Powering networks on chips*, Proc. of ISSS, 2001, pp. 33–38.

[20] L. Benini and G. De Micheli, *Networks on chips: A new SoC paradigm*, Computer **35** (2002), pp. 70–78.

[21] D. Bertozzi and L. Benini, *Xpipes: A network-on-chip architecture for gigascale systems-on-chip*, IEEE Circuits and Systems Magazine **4** (2004), no. 2, pp. 18–31.

[22] D.A. Bini, B. Meini, S. Steffé, and B. Van Houdt, *Structured Markov chains solver: Algorithms*, Proc. of SMCTools, 2006.

[23] D.A. Bini, B. Meini, S. Steffé, and B. Van Houdt, *Structured Markov chains solver: Software tools*, Proc. of SMCTools, 2006.

[24] C. Bisdikian, *A note on the conservation law for queues with batch arrivals*, IEEE Transactions on Communications **41** (1993), no. 6, pp. 832–835.

[25] T. Bjerregaard and S. Mahadevan, *A survey of research and practices of Network-on-chip*, ACM Computing Surveys **38** (2006), no. 1, pp. 1–51.

[26] J.P.C. Blanc, *A numerical approach to cyclic-service queueing models*, Queueing Systems **6** (1990), no. 1, pp. 173–188.

[27] J.P.C. Blanc, *The power-series algorithm applied to cyclic polling systems*, Stochastic Models **7** (1991), no. 4, pp. 527–545.

[28] J.P.C. Blanc, *An algorithmic solution of polling models with limited service disciplines*, IEEE Transactions on Communications **40** (1992), no. 7, pp. 1152–1155.

[29] J.P.C. Blanc, *Performance evaluation of polling systems by means of the power-series algorithm*, Annals of Operations Research **35** (1992), no. 3, pp. 155–186.

[30] N. Boot, *Throughput and delay analysis for a single router in networks on chip*, Master's thesis, Eindhoven University of Technology, June 2005.

[31] O.J. Boxma and W.P. Groenendijk, *Two queues with alternating service and switching times*, Queueing Theory and its Applications: Liber Amicorum for J.W. Cohen (O.J. Boxma and R. Syski, eds.), North-Holland, Amsterdam, 1988, pp. 261–282.

[32] O.J. Boxma and W.P. Groenendijk, *Waiting times in discrete-time cyclic-service systems*, IEEE Transactions on Communications **36** (1988), no. 2, pp. 164–170.

[33] O.J. Boxma and B.W. Meister, *Waiting-time approximations for cyclic-service systems with switch-over times*, ACM SIGMETRICS Performance Evaluation Review **14** (1986), no. 1, pp. 254–262.

[34] O.J. Boxma and B.W. Meister, *Waiting-time approximations in multi-queue systems with cyclic service*, Performance Evaluation **7** (1987), no. 1, pp. 59–70.

[35] O.J. Boxma and G.J. van Houtum, *The compensation approach applied to a 2x2 switch*, Probability in the Engineering and Informational Sciences **7** (1993), pp. 171—193.

[36] O.J. Boxma and J.A. Weststrate, *Waiting times in polling systems with Markovian server routing*, Messung, Modellierung und Bewertung von Rechensystemen und Netzen (G. Stiege and J.S. Lie, eds.), Springer-Verlag, Berlin, 1989, pp. 89–105.

[37] H. Bruneel and B.G. Kim, *Discrete Time Models for Communication Systems including ATM*, Kluwer Academic Publishers, Norwell, 1993.

[38] R.K.C. Chang and S. Lam, *A novel approach to queue stability analysis of polling models*, Performance Evaluation **40** (2000), no. 1, pp. 27–46.

[39] J.S. Chen and T.E. Stern, *Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch*, IEEE Journal on Selected Areas in Communications **9** (1991), no. 3, pp. 439–449.

[40] S.-T. Chuang, A. Goel, N.W. McKeown, and B. Prabhakar, *Matching output queueing with a combined input/output-queued switch*, IEEE Journal on Selected Areas in Communications **17** (1999), no. 6, pp. 1030–1039.

[41] J.W. Cohen, *On the determination of the stationary distribution of a symmetric clocked buffered switch*, CWI report BS-R9427, 1994.

[42] J.W. Cohen, *On the asymmetric clocked buffered switch*, Queueing Systems **30** (1998), no. 3, pp. 385–404.

[43] J.W. Cohen and O.J. Boxma, *The M/G/1 queue with alternating servic formulated as a Riemann-Hilbert problem*, Performance '81 (F.J. Kylstra, ed.), North-Holland, Amsterdam, 1981, pp. 181–199.

[44] J.W. Cohen and O.J. Boxma, *Boundary Value Problems in Queueing System Analysis*, North-Holland, Amsterdam, 1983.

[45] J.G. Dai and J.M. Harrison, *The QNET method for two-moment analysis of closed manufacturing systems*, The Annals of Applied Probability **3** (1993), no. 4, pp. 968–1012.

[46] J.G. Dai and B. Prabhakar, *The throughput of data switches with and without speedup*, Proc. of IEEE INFOCOM, 2000, pp. 556–564.

[47] Y. Dallery and Y. Frein, *On decomposition methods for tandem queueing networks with blocking*, Operations Research **41** (1993), no. 2, pp. 386–399.

[48] Y. Dallery and S.B. Gershwin, *Manufacturing flow line systems: A review of models and analytical results*, Queueing Systems **12** (1992), no. 1, pp. 3–94.

[49] W.J. Dally and B. Towles, *Route packets, not wires: On-chip interconnection networks*, Proc. of the Design Automation Conference, 2001, pp. 684–689.

[50] B. Desert and H. Daduna, *Discrete time tandem networks of queues: Effects of different regulation schemes for simultaneous events*, Performance Evaluation **47** (2002), no. 2, pp. 73–104.

[51] H. Dror and U. Yechiali, *Closed polling models with failing nodes*, Queueing Systems **35** (2000), no. 1, pp. 55–81.

[52] M. Eisenberg, *Two queues with alternating service*, SIAM Journal of Applied Mathematics **36** (1979), no. 2, pp. 287–303.

[53] G. Fayolle and R. Iasnogorodski, *Two coupled processors: The reduction to a Riemann-Hilbert problem*, Probability Theory and Related Fields **47** (1979), no. 3, pp. 325–351.

[54] G. Fayolle, V.A. Malyshev, and M.V. Menshikov, *Topics in the Constructive Theory of Countable Markov Chains*, Cambridge University Press, 1995.

[55] W. Feng, M. Kowada, and K. Adachi, *A two-queue model with Bernoulli service schedule and switching times*, Queueing Systems **30** (1998), no. 3–4, pp. 405–434.

[56] L. Flatto and H.P. McKean, *Two queues in parallel*, Communications on Pure and Applied Mathematics **30** (1977), no. 2, pp. 255–263.

[57] H.D. Friedman, *Reduction methods for tandem queueing systems*, Operations Research **13** (1965), no. 1, pp. 121–131.

[58] I. Frigui and A.S. Alfa, *Analysis of a discrete time table polling system with MAP input and time-limited service discipline*, Telecommunication Systems **12** (1999), no. 1, pp. 51–77.

[59] S.W. Fuhrmann and Y.T. Wang, *Analysis of cyclic service systems with limited service: Bounds and approximations*, Performance Evaluation **9** (1988), no. 1, pp. 35–54.

[60] M. Garetto, R.L. Cigno, M. Meo, and M. Ajmone Marsan, *A detailed and accurate closed queueing network model of many interacting TCP flows*, Proc. of IEEE INFOCOM, 2001, pp. 1706–1715.

[61] M. Gerla and L. Kleinrock, *Flow control: A comparative study*, IEEE Transactions on Communications **28** (1980), no. 4, pp. 553–574.

[62] S. González Pestana, E. Rijpkema, A. Rădulescu, K. Goossens, and O.P. Gangwal, *Cost-performance trade-offs in networks on chips: A simulation based approach*, Proc. of DATE, 2004, pp. 764–769.

[63] K. Goossens, J. Dielissen, J. Van Meerbergen, P. Poplavko, A. Rădulescu, E. Rijpkema, E. Waterlander, and P. Wielage, *Guaranteeing the quality of services in networks on chip*, Networks on Chips (A. Jantsch and H. Tenhunen, eds.), Kluwer, Dordrecht, 2003, pp. 61–82.

[64] K. Goossens, J. Dielissen, and A. Rădulescu, *Æthereal network on chip: Concepts, architectures, and implementations*, IEEE Journal on Design & Test of Computers **22** (2005), no. 5, pp. 414–421.

[65] K. Goossens, L. Mhamdi, and I.V. Senín, *Internet-router buffered crossbars based on networks on chip*, Proc. of Euromicro Symposium on DSD.

[66] K. Goossens and A. Rădulescu, *Communication services for networks on chip*, Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation (S.S. Bhattacharyya, E.F. Deprettere, and J. Teich, eds.), Marcel Dekker, 2004, pp. 193–213.

[67] A. Gravey and G. Hébuterne, *Simultaneity in discrete-time single server queues with Bernoulli inputs*, Performance Evaluation **14** (1992), no. 2, pp. 123–131.

[68] W.P. Groenendijk and H. Levy, *Performance analysis of transaction driven computer systems via queueing analysis of polling models*, IEEE Transactions on Computers **41** (1992), no. 4, pp. 455–466.

[69] P. Guerrier and A. Greiner, *A generic architecture for on-chip packet-switched interconnections*, Proc. of DATE, 2000, pp. 250–256.

[70] A.K. Gupta and N.D. Georganas, *Analysis of a packet switch with input and output buffers and speed constraints*, Proc. of IEEE INFOCOM, 1991, pp. 694–700.

[71] A. Hansson, K. Goossens, and A. Rădulescu, *A unified approach to constrained mapping and routing on network-on-chip architectures*, Proc. of CODES+ISSS, 2005, pp. 75–80.

[72] O.C. Ibe and X. Cheng, *Stability conditions for multiqueue systems with cyclic service*, IEEE Transactions on Automatic Control **33** (1988), no. 1, pp. 102–103.

[73] I. Iliadis and W.E. Denzel, *Analysis of packet switches with input and output queuing*, IEEE Transactions on Communications **41** (1993), no. 5, pp. 731–740.

[74] S. Jaffe, *Equilibrium results for a pair of coupled discrete-time queues*, Ultracomputer Note, NYA Ultracomputer Research Lab, Courant Institute of Mathematical Sciences, New York, 1989.

[75] S. Jaffe, *The equilibrium distribution for a clocked buffered switch*, Probability in the Engineering and Informational Sciences **6** (1992), pp. 425–438.

[76] M.J. Karol, K.Y. Eng, and H. Obara, *Improving the performance of input-queued ATM packet switches*, Proc. of IEEE INFOCOM, 1992, pp. 110–115.

[77] M.J. Karol, M.G. Hluchyj, and S.P. Morgan, *Input versus output queueing on a space-division packet switch*, IEEE Transactions on Communications **35** (1987), no. 12, pp. 1347–1356.

[78] J. Keilson and L.D. Servi, *Oscillating random walk models for GI/G/1 vacation systems with Bernoulli schedules*, Journal of Applied Probability **23** (1986), no. 3, pp. 790–802.

[79] H. Kim and K. Kim, *Performance analysis of the multiple input-queued packet switch with the restricted rule*, IEEE/ACM Transactions on Networking **11** (2003), no. 3, pp. 478–487.

[80] H. Kim, K. Kim, and Y. Lee, *Derivation of the mean cell delay and cell loss probability for multiple input-queued switches*, IEEE Communications Letters **4** (2000), no. 4, pp. 140–142.

[81] J.F.C. Kingman, *Two similar queues in parallel*, The Annals of Mathematical Statistics (1961), no. 4, pp. 1314–1323.

[82] L. Kleinrock, *Queueing Systems. Volume I: Theory*, John Wiley & sons, New York, 1975.

[83] L. Kleinrock, *Queueing Systems. Volume II: Computer Applications*, John Wiley & sons, New York, 1976.

[84] L. Kleinrock and H. Levy, *The analysis of random polling systems*, Operations Research **36** (1988), no. 5, pp. 716–732.

[85] C. Kolias and L. Kleinrock, *The odd-even input-queueing ATM switch: Performance evaluation*, Proc. of IEEE ICC, 1996, pp. 1674–1679.

[86] C. Kolias and L. Kleinrock, *Throughput analysis of multiple input-queueing in ATM switches*, Proc. of Broadband Communications (L. Mason and A. Casaca, eds.), Chapman & Hall, London, 1996, pp. 382–393.

[87] D.D. Kouvatsos and N.P. Xenios, *Maximum entropy analysis of general queueing networks with blocking*, First International Conference on Queueing Networks with Blocking (H.G. Perros and T. Altiok, eds.), 1989, pp. 281–309.

[88] D.D. Kouvatsos and N.P. Xenios, *MEM for arbitrary queueing networks with multiple general servers and repetitive-service blocking*, Performance Evaluation **10** (1989), no. 3, pp. 169–195.

[89] P. Krishna, N.S. Patel, A. Charny, and R.J. Simcoe, *On the speedup required for work-conserving crossbar switches*, IEEE Journal on Selected Areas in Communications **17** (1999), no. 6, pp. 1057–1066.

[90] K. Lahiri, A. Raghunathan, and S. Dey, *Design space exploration for optimizing on-chip communication architectures*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **23** (2004), no. 6, pp. 952–961.

[91] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modeling*, SIAM, Philadelphia, 1999.

[92] D-S. Lee and B. Sengupta, *An approximate analysis of a cyclic server queue with limited service and reservations*, Queueing Systems **11** (1992), no. 1–2, pp. 153–178.

[93] H.G. Lee, N. Chang, U.Y. Ogras, and R. Marculescu, *On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches*, ACM Transactions on Design Automation of Electronic Systems **12** (2007), no. 3, pp. 1–20.

[94] K.K. Leung, *Cyclic-service systems with probabilistically-limited service*, IEEE Journal on Selected Areas in Communications **9** (1991), no. 2, pp. 185–193.

[95] S.-Q. Li, *Nonuniform traffic analysis on a nonblocking space-division packet switch*, IEEE Transactions on Communications **38** (1990), no. 7, pp. 1085–1096.

[96] S.-Q. Li, *Performance of a nonblocking space-division packet switch with correlated input traffic*, IEEE Transactions on Communications **40** (1992), no. 1, pp. 97–108.

[97] M. Mandjes and M. van Uitert, *Sample-path large deviations for tandem and priority queues with Gaussian inputs*, Annals of Applied Probability **15** (2005), no. 2, pp. 1193–1226.

[98] N.W. McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*, Ph.D. thesis, Univ. California, Berkeley, 1995.

[99] N.W. McKeown, *The iSLIP scheduling algorithm for input-queued switches*, IEEE/ACM Transactions on Networking **7** (1999), no. 2, pp. 188–201.

[100] N.W. McKeown and T.E. Anderson, *A quantitative comparison of iterative scheduling algorithms for input-queued switches*, Computer Networks and ISDN Systems **30** (1998), no. 24, pp. 2309–2326.

[101] N.W. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, *Achieving 100% throughput in an input-queued switch*, IEEE Transactions on Communications **47** (1999), no. 8, pp. 1260–1267.

[102] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, *The Nostrum backbone - a communication protocol stack for networks on chip*, Proc. of VLSID, 2004, pp. 693–696.

[103] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, *HERMES: An infrastructure for low area overhead packet-switching networks on chip*, Integration, the VLSI Journal **38** (2004), no. 1, pp. 69–93.

[104] J.A. Morrison, *A combinatorial lemma and its application to concentrating trees of discrete-time queues*, The Bell Systems Technical Journal **57** (1978), no. 5, pp. 1645–1652.

[105] M.J. Neely, C.E. Rohrs, and E. Modiano, *Equivalent models for queueing analysis of deterministic service time tree networks*, IEEE Transactions on Information Theory **51** (2005), no. 10, pp. 1–10.

[106] M.F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and their Applications*, Probability: Pure and Applied, vol. 5, Marcel Dekker Inc., New York, 1989.

[107] M.F. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, Dover Publications, New York, 1994.

[108] U.Y. Ogras and R. Marculescu, *Analytical router modeling for networks-on-chip performance analysis*, Proc. of DATE, 2007, pp. 1096–1101.

[109] Y. Oie, M. Murata, K. Kubota, and H. Miyahara, *Performance analysis of nonblocking packet switch with input and output buffers*, IEEE Transactions on Communications **40** (1992), no. 8, pp. 1294–1297.

[110] R.O. Onvural, *Survey of closed queueing networks with blocking*, ACM Computing Surveys **22** (1990), no. 2, pp. 83–121.

[111] A. Pattavina and G. Bruzzi, *Analysis of input and output queueing for nonblocking ATM switches*, IEEE/ACM Transactions on Networking **1** (1993), no. 3, pp. 314–328.

[112] H.G. Perros, *Queueing Networks with Blocking*, Oxford University Press, New York, 1994.

[113] B. Prabhakar and N.W. McKeown, *On the speedup required for combined input- and output-queued switching*, Automatica **35** (1999), no. 12, pp. 1909–1920.

[114] A. Rădulescu, J. Dielissen, K. Goossens, E. Rijpkema, and P. Wielage, *An efficient on-chip network interface offering guaranteed services, shared memory abstraction, and flexible network configuration*, Proc. of DATE, 2004, pp. 878–883.

[115] M.I. Reiman and L.M. Wein, *Heavy traffic analysis of polling systems in tandem*, Operations Research **47** (1999), no. 4, pp. 524–534.

[116] M. Reiser, *A queueing network analysis of computer communication networks with window flow control*, IEEE Transactions on Communications **27** (1979), no. 8, pp. 1199–1209.

[117] J.A.C. Resing, *Polling systems and multitype branching processes*, Queueing Systems **13** (1993), no. 4, pp. 409–426.

[118] J.A.C. Resing and L. Örmeci, *A tandem queueing model with coupled processors*, Operations Research Letters **31** (2003), no. 5, pp. 383–389.

[119] S.I. Resnick, *Adventures in Stochastic Processes*, 2nd ed., Birkhäuser, Boston, 1994.

[120] E. Rijpkema, K. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander, *Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip*, IEE Proceedings Computers and Digital Techniques **150** (2003), no. 5, pp. 294–302.

[121] I. Rubin, *Communication networks: Message path delays*, IEEE Transactions on Information Theory **20** (1974), no. 6, pp. 738–745.

[122] I. Rubin, *Message path delays in packet-switching communication networks*, IEEE Transactions on Communications **23** (1975), no. 2, pp. 186–192.

[123] I. Rubin, *An approximate time-delay analysis for packet-switching communication networks*, IEEE Transactions on Communications **24** (1976), no. 2, pp. 210–222.

[124] I. Rubin and L.F.M. de Moraes, *Message delay analysis for polling and token multiple-access schemes for local communication networks*, IEEE Journal on Selected Areas in Communications **1** (1983), no. 5, pp. 935–947.

[125] D. Shah and D. Wischik, *Optimal scheduling algorithms for input-queued switches*, Proc. of IEEE INFOCOM (2006), pp. 1–11.

[126] M.S. Shalmon, *Exact delay analysis of packet-switching concentrating networks*, IEEE Transactions on Communications **35** (1987), no. 12, pp. 1265–1271.

[127] M.S. Shalmon and M.A. Kaplan, *A tandem network of queues with deterministic service and intermediate arrivals*, Operations Research **32** (1984), no. 4, pp. 753–773.

[128] M.M. Srinivasan, *An approximation for mean waiting times in cyclic server systems with nonexhaustive service*, Performance Evaluation **9** (1988), no. 1, pp. 17–33.

[129] M.M. Srinivasan, *Non-deterministic polling systems*, Management Science **37** (1991), no. 6, pp. 667–681.

[130] S. Stidham jr., *A last word on $L = \lambda W$*, Operations Research **22** (1974), no. 2, pp. 417–421.

[131] R. Suri and G.W. Diehl, *A variable buffer-size model and its use in analyzing closed queueing networks with blocking*, Management Science **32** (1986), no. 2, pp. 206–224.

[132] H. Takagi, *Analysis of Polling Systems*, MIT Press, Cambridge, Massachusetts, 1986.

[133] H. Takagi, *Queueing analysis of polling models: An update*, Stochastic Analysis of Computer and Communication Systems (H. Takagi, ed.), North-Holland, 1990, pp. 267–318.

[134] H. Takagi, *Queueing Analysis: A Foundation of Performance Evaluation*, vol. 3: Discrete-time systems, North-Holland, Amsterdam, 1993.

[135] H. Takagi, *Queueing analysis of polling models: Progress in 1990-1994*, Frontiers in Queueing: Models and Applications in Science and Engineering (J.H. Dshalalow, ed.), CRC Press, Boca Raton, 1997, pp. 119–146.

[136] Y. Tamir and G.L. Frazier, *High-performance multiqueue buffers for VLSI communication switches*, Computer Architecture News **16** (1988), no. 2, pp. 343–354.

[137] T.E. Tedijanto, *Exact results for the cyclic-service queue with a Bernoulli schedule*, Performance Evaluation **11** (1990), no. 2, pp. 107–115.

[138] J.S.H. van Leeuwaarden, *Queueing Models for Cable Access Networks*, Ph.D. thesis, Eindhoven University of Technology, 2005.

[139] J.S.H. van Leeuwaarden and J.A.C. Resing, *A tandem queue with coupled processors: Computational issues*, Queueing Systems **51** (2005), no. 1–2, pp. 29–52.

[140] M. van Vuuren and E.M.M. Winands, *Iterative approximation of k-limited polling systems*, Queueing Systems **55** (2007), no. 3, pp. 161–178.

[141] G.V. Varatkar and R. Marculescu, *On-chip traffic modeling and synthesis for MPEG-2 video applications*, IEEE Transactions on VLSI Systems **12** (2004), no. 1, pp. 108–119.

[142] V.M. Vishnevskii and O.V. Semenova, *Mathematical methods to study the polling systems*, Automation and Remote Control **67** (2006), no. 2, pp. 173–220.

[143] J.A. Weststrate, *Analysis and Optimization of Polling Models*, Ph.D. thesis, Tilburg University, 1992.

[144] J.A. Weststrate and R.D. van der Mei, *Waiting times in a two-queue model with exhaustive and Bernoulli service*, Mathematical Methods of Operations Research **40** (1994), no. 3, pp. 289–303.

[145] W. Whitt, *A review of $L = \lambda W$ and extensions*, Queueing Systems **9** (1991), pp. 235–268.

[146] P. Wielage, E.J. Marinissen, M. Altheimer, and C. Wouters, *Design and DfT of a high-speed area-efficient embedded asynchronous FIFO*, Proc. of DATE, 2007, pp. 853–858.

[147] E.M.M. Winands, *Mean value analysis for polling systems*, Queueing Systems **54** (2006), no. 1, pp. 35–44.

[148] R.W. Wolff, *Poisson arrivals see time averages*, Operations Research **30** (1982), no. 2, pp. 223–231.

[149] Y. Xiong and H. Bruneel, *Buffer contents and delay for statistical multiplexers with fixed-length packet-train arrivals*, Performance Evaluation **17** (1993), no. 1, pp. 31–42.

[150] D.D. Yao and J.A. Buzacott, *The exponentialization approach to flexible manufacturing system models with general processing times.*, European Journal of Operational Research **24** (1986), no. 3, pp. 410–416.

[151] K. Yoshigoe and K.J. Christensen, *An evolution to crossbar switches with virtual output queuing and buffered cross points*, IEEE Network **17** (2003), no. 5, pp. 48–56.

Summary

Performance Analysis of Networks on Chips

Modules on a chip (such as processors and memories) are traditionally connected through a single link, called a *bus*. As chips become more complex and the number of modules on a chip increases, this connection method becomes inefficient because the bus can only be used by one module at a time.

Networks on chips are an emerging technology for the connection of on-chip modules. In networks on chips, *switches* are used to transmit data from one module to another, which entails that multiple links can be used simultaneously so that communication is more efficient.

Switches consist of a number of input ports to which data arrives and output ports from which data leaves. If data at multiple input ports has to be transmitted to the same output port, only one input port may actually transmit its data, which may lead to congestion.

Queueing theory deals with the analysis of congestion phenomena caused by competition for service facilities with scarce resources. Such phenomena occur, for example, in traffic intersections, manufacturing systems, and communication networks like networks on chips. These congestion phenomena are typically analysed using stochastic models, which capture the uncertain and unpredictable nature of processes leading to congestion (such as irregular car arrivals to a traffic intersection). Stochastic models are useful tools for the analysis of networks on chips as well, due to the complexity of data traffic on these networks. In this thesis, we therefore study queueing models aimed at networks on chips.

The thesis is centred around two key models: A model of a switch in isolation, the so-called single-switch model, and a model of a network of switches where all traffic has the same destination, the so-called network of polling stations. For both models we are interested in the *throughput* (the amount of data transmitted per time unit) and the mean *delay* (the time it takes data to travel across the network).

Single-switch models are often studied under the assumption that the number of ports tends to infinity and that traffic is uniform (i.e., on average equally many packets arrive to all buffers, and all possible destinations are equally likely). In networks on chips, however, the number of buffers is typically small. We introduce a new approximation specifically aimed at small switches with (memoryless) Bernoulli arrivals. We show that, for such switches, this approximation is more accurate than currently known approximations.

As traffic in networks on chips is usually *non-uniform*, we also extend our approximation to non-uniform switches. The key difference between uniform and non-uniform switches is that in non-uniform switches, all queues have a different maximum throughput. We obtain a very accurate approximation of this throughput, which allows us to extend the mean delay approximation. The extended approximation is derived for Bernoulli arrivals and correlated arrival processes. Its accuracy is verified through a comparison with simulation results.

The second key model is that of concentrating tree networks of polling stations (polling stations are essentially switches where all traffic has the same output port as destination). *Single* polling stations have been studied extensively in literature, but only few attempts have been made to analyse *networks* of polling stations. We establish a reduction theorem that states that networks of polling stations can be reduced to single polling stations while preserving some information on mean waiting times. This reduction theorem holds under the assumption that the last node of the network uses a so-called *HoL-based* service discipline, which means that the choice to transmit data from a certain buffer may only depend on which buffers are empty, but not on the amount of data in the buffers.

The reduction theorem is a key tool for the analysis of networks of polling stations. In addition to this, mean waiting times in single polling stations have to be calculated, either exactly or approximately. To this end, known results can be used, but we also devise a new single-station approximation that can be used for a large subclass of HoL-based service disciplines.

Finally, networks on chips typically implement flow control, which is a mechanism that limits the amount of data in the network from one source. We analyse the division of throughput over several sources in a network of polling stations with flow control. Our results indicate that the throughput in such a network is determined by an interaction between buffer sizes, flow control limits, and service disciplines. This interaction is studied in more detail by means of a numerical analysis.

About the author

Paul Beekhuizen was born in Rotterdam on April 15th of 1982. Paul attended high school at the Jacob-Roelandslyceum in Boxtel, where he graduated in 2000. After that he studied industrial and applied mathematics at Eindhoven University of Technology, where he obtained his Bachelor degree in 2003 and his Master degree in 2005 (both cum laude). For the final project of his Master program, Paul did research on the performance analysis of networks on chips. This research was performed at Philips Research, under the supervision of Dee Denteneer (Philips Research) and Bert Zwart (Eindhoven University of Technology).

In December 2005, Paul started as a PhD student at Philips Research. During his four year period as a PhD student, he had an additional affiliation with EURANDOM. His supervisors were Dee Denteneer (Philips Research) and Jacques Resing (Eindhoven University of Technology) and his promotor is Onno Boxma (Eindhoven University of Technology).