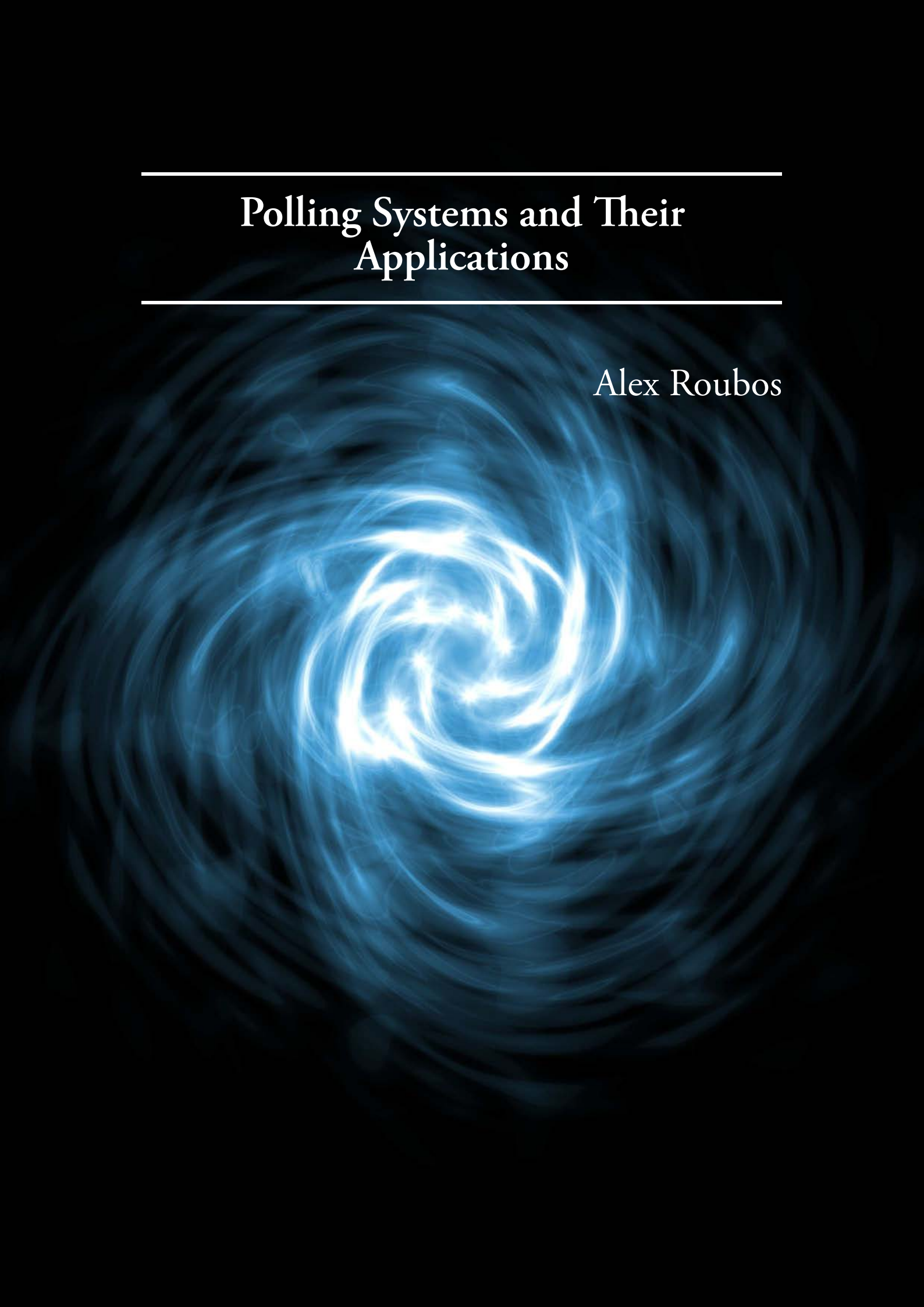
Polling Systems and Their Applications

Alex Roubos



vrije Universiteit *amsterdam*
Business Mathematics and Informatics

Polling Systems and Their Applications

Alex Roubos
aroubos@cs.vu.nl

October 24, 2007

Preface

This thesis is part of acquiring the Master's degree in Business Mathematics and Informatics. Business Mathematics and Informatics is a multidisciplinary programme, aimed at improving business processes by applying a combination of methods based upon mathematics, computer science and business management. These three disciplines will also play a central role throughout this thesis.

In this thesis, I will give a basic introduction to polling systems together with a couple of things related to them, such as their applications. Polling systems belong to an important class of queueing systems, because a wide variety of applications lend themselves to be modeled as a polling system, thereby opening possibilities for optimization purposes. Because of this, the business aspect of Business Mathematics and Informatics is present in the subject of polling systems. Also the mathematical and computer science aspects are clearly present since polling systems are analyzed by means of mathematical methods, usually resulting in a certain computational scheme. The results of this can then be computed numerically.

After reading this thesis, one is able to formulate an answer to the following questions: "What are polling systems?", "How can polling systems be applied to real-life situations?" and "Which techniques exist to analyze polling systems?".

Finally, I would like to thank my supervisor prof. dr. Rob van der Mei for his help and support and for acquainting me with the subject of polling systems.

Alex Roubos
Aalsmeer, 2007

Summary

A queueing model is a mathematical model describing situations in which customers request for service from a server. Customers to whom service cannot be rendered immediately take place in a queue. The classical polling model is a special class of queueing models consisting of multiple queues and a single server that visits the queues in some order to serve the customers waiting at the queues. The service policy determines which customers are served during the visit of the server to a queue. The routing scheme determines in which order the server visits the queues. The server typically incurs some amount of switch-over time to proceed from one queue to the next.

Polling systems are applied for the analysis of situations in which different types of user require service from a common server. In this way polling systems can for instance be applied to the modeling and performance evaluation of computer systems, communication networks, traffic systems, flexible manufacturing and production systems.

The analysis of polling systems is very difficult. Most polling systems are not even exact analyzable with the help of existing mathematical techniques. And even if a polling system allows an exact analysis, the analysis still does not always lead to manageable expressions for performance measures such as mean waiting times. Because of this, several numerical techniques have been developed. Numerical techniques, unlike analytical methods, do not give an exact expression for the performance measures as a function of the system parameters, but they can be used to numerically compute performance measures for a given system. The buffer occupancy method and the descendant set approach are two most well-known numerical techniques.

A different approach for computing performance measures of all kinds of models is simulation. Simulation can be applied to all polling systems, including systems for which no numerical algorithms exist. For these systems, simulation is the only possibility for obtaining these performance measures. In spite of their enormous flexibility, simulation comes with two disadvantages. In many cases simulation techniques are rather inefficient and the results based on simulation are relatively inaccurate compared with numerical algorithms. The latter can be partially solved by simulating for a longer time period.

Contents

Chapter 1

Introduction

Queueing arises on a daily basis in many real-life situations. Waiting lines occur at post offices, in traffic situations and at elevators, but also on a more abstract level in communication systems and computer networks. The undesirability of congestion has raised the need to reach a better understanding of queueing situations. For this purpose queueing models are developed. The main entities in queueing models are *customers* who arrive at a station requiring service from a *server*. Arriving customers to whom service cannot be rendered immediately take place in a *queue*. Typical performance measures are means and standard deviations of waiting times and queue lengths. This thesis is devoted to a special class of queueing models called *polling models*.

A polling system is a multi-queue single-server system in which the server visits the queues in some order to serve the customers waiting at the queues, typically incurring some amount of switch-over time to proceed from one queue to the next. Polling models occur naturally in the modeling of systems in which service capacity (e.g., CPU, bandwidth, manpower) is shared among different types of user, each type having specific traffic characteristics and performance requirements. Typical application areas of polling models are computer communication systems, logistics, flexible manufacturing systems, production systems and maintenance systems [25, 44]. Over the past few decades the performance analysis of polling models has received much attention in the literature (cf., e.g., [41, 42, 43, 45] for some early work and [47] for a recent paper).

The term *polling* originates from the so-called polling data link control scheme, in which a central computer interrogates each terminal on a communication line to determine whether or not it has data to transmit. The addressed terminal transmits data and the computer then switches to the next terminal. Here, the server represents the computer and a queue corresponds to a terminal [46].

A polling system can be expressed in a couple of parameters. Standard parameters for queueing systems are the number of queues, which is usually denoted by N, and the traffic characteristics at the queues, which consist of the arrival process and the service process. With polling systems the switch-over process between the queues is introduced together

with two other important parameters: the *service policy* and the *routing scheme*.

The service policy or service discipline specifies how many customers are served during the visit of the server to a queue. The most common service policies are the *exhaustive* service policy, under which the server continues to work until the queue has become empty, and the *gated* service policy, under which exactly those customers are served who were present at the moment when the server arrived. A whole abundance of service policies can be created by putting some cut-off mechanism on the classical exhaustive and gated service policies (cf., Section 3.2). A special possibility is *mixed* service, where not all queues are served according to the same service policy.

The routing scheme determines the order in which the queues are served. A distinction can be made between *static* and *dynamic* routing. Static routing is independent of the state of the system. The traditional routing scheme is the *cyclic* routing, which is in the order $(1, 2, \ldots, N)$. To prioritize certain queues, cyclic routing has been extended to *periodic* routing, in which the server visits the queues periodically according to a polling table. Typical examples are *elevator* routing, which is in the order $(1, 2, \ldots, N-1, N, N, N-1, \ldots, 2, 1)$, and *star* routing, which is in the order $(1, 2, 1, 3, \ldots, 1, N)$. Dynamic routing is state-dependent routing. The decision of the server as to which queue to visit may depend on some information available to the server, such as queue lengths. For instance, it might not make sense to move to an empty queue while customers are waiting at other queues. An example is the *serve-longest-queue* policy. A disadvantage of dynamic routing is the possibility of having a huge state space dimension, resulting in an unfeasible computation time. This is called the *curse of dimensionality*.

1.1 Research objectives

This thesis is concerned with a relatively small literature study of polling systems. It is by no means meant to give a complete overview, but rather it should be considered as an introductory survey consisting of the most important and basic results only. There are many applications that allow themselves to be modeled as a polling system. The first objective of this thesis is therefore **to identify the application areas of polling systems** and **to give examples of how these applications can be modeled as polling systems**.

Once applications are fit into the mathematical polling model, performance measures are to be computed. This is no easy task as illustrated by the hundreds of research papers solely devoted to this extend, and the results that are available can only be applied to a very limited class of polling systems. The second research objective concerns the analysis of polling systems. More specifically, this objective is **to research the available techniques to analyze polling systems**.

Polling systems that cannot be analyzed by means of analytical methods or numerical techniques are very unfortunate. Especially for such systems, simulation comes to the salvation as simulation is then the only possibility for obtaining performance measures. At

the same time, all kinds of derivatives of these performance measures can be obtained in the same run requiring almost no extra effort at all. For these reasons, the third research objective of this thesis is **to develop a simulation program able to simulate a wide variety of polling systems**.

The remainder of this thesis is organized as follows. In Chapter 2 the applications of polling systems are discussed. In here, a number of application areas are elaborated in further detail and it is explained how these applications can be modeled as polling systems. The mathematical analysis of polling systems is the subject of Chapter 3. An overview of some successful techniques to analyze the performance of polling systems is given. Chapter 4 is about simulation. In here, the simulation program that is devised to simulate polling systems is explained, and certain related aspects are discussed.

Chapter 2

Applications

Polling has been used as early as in the late 1950s, involving a patrolling machine repairman who inspects a number of machines to check whether a breakdown has occurred and, if so, eliminates such breakdowns (cf., [27, 28]). Later, polling systems were used to study the problem of vehicle-actuated traffic signal control (cf., [35, 36]). The introduction of computer communication networks in the 1970s has created a wide array of problems. Initially, polling was used for data transfer from terminals on a communication line to a central computer (cf., [18]). Later, it was used for token passing protocols in local area networks (cf., [5]). In the application to computer systems, polling systems were used for the scanning mechanism of the hard disk drive, and for load sharing in multiprocessor computers. Other applications of polling systems are found, among others, in transportation networks (cf., [3]), public transportation systems (cf., [10]), shipyard loading (cf., [9]), videotex (cf. [26]), mail delivery (cf., [34]) and elevators (cf., [15, 16]).

Before some of the successful applications are discussed, let's first take a moment to emphasize the importance of polling systems. Waiting time is a critical issue for problems dealing with passengers (e.g., traffic lights and elevators) while queue lengths (e.g., storage room) are more critical for problems dealing with freight. To this effect, it is very helpful to know the moments of these performance measures beforehand. For example, the moments of the queue lengths will help determine the size of a physical buffer. Additionally, a clear understanding of polling systems will allow designers to optimize overall system performance and improve efficiency by, e.g., reducing wasted resources and reducing complexity of the network.

2.1 Communication networks

Takagi [44] showed how some of the results of polling systems can be used for the modeling and performance evaluation of communication networks. Three of these successful applications are highlighted in this section. The main purpose of communication networks is

to facilitate the exchange of information between two entities. The information (e.g., files and e-mail) is put into packets conforming to the network protocols and then sent over the transmission medium. Within a network, users compete to have access to the transmission medium.

2.1.1 Half duplex transmission

Half duplex transmission is a mode of transmitting data between two parties on a shared communication line. Transmission is possible in either direction but not in both directions simultaneously. (A similar situation can be observed in everyday life, e.g., a traffic light at the intersection of two one-way streets, a narrow bridge or passage, conversation with a walkie-talkie and other push-to-talk services.) Suppose that a central computer and a communication control unit connected to several data terminals exchange messages over a half duplex line. When the transmission from the communication control unit is complete, a finite time is needed to reverse the direction of transmission on the line. Output messages are sent from the computer to the communication control unit, which delivers them to the terminals. After receiving a polling message from the computer and again reversing the direction of transmission, the communication control unit can start sending input messages to the computer, and this cycle is repeated.

In the queueing model of this system, customers correspond to the input and output messages, and the server represents the communication line between the computer and the communication control unit that allows the alternating transmission of messages. Denote by queue 1 the computer and by queue 2 the communication control unit. The service time at queue 1 is the transmission time of an output message, and the service time at queue 2 is the transmission time of an input message. The switch-over time from queue 1 to queue 2 consists of the time for sending the polling message and the facility reversal time. The switch-over time from queue 2 to queue 1 consists only of the facility reversal time.

2.1.2 Polling data link control

Polling control has often been employed in network configurations in which geographically dispersed terminals are connected to a central computer in a tree topology or a loop topology. There are two types of polling control. In *roll-call polling*, the computer has a polling sequence table according to which it interrogates each terminal. The addressed terminal then transmits all waiting messages to the computer. When the transmission from one terminal is complete, the computer starts polling the next terminal. In the polling sequence table, the network designer can order terminals in exact cyclic order, or in any sequence and frequency to prioritize terminals. Roll-call polling is suitable for a tree topology. In a loop topology, on the other hand, *hub polling* is often used. In this case, a natural polling sequence is determined by the position of terminals on the loop. The central computer initiates polling by interrogating the terminal at the end of the loop. This terminal transmits

its waiting messages, to which it appends a polling message for the next downstream (in the direction of transmission) terminal. The latter terminal similarly adds its own messages followed by another polling message, and so on. At the completion of a polling cycle, the central computer collects all the messages and assumes control.

In polling models the central computer is represented by the server, the terminals are represented by the queues and the messages of these terminals are represented by the customers. The switch-over time is the time between polling adjacent terminals and depends on the network topology and whether roll-call or hub polling is used. The service times consist of the message length and the line speed.

2.1.3 Token ring network

A local area network consists of a number of stations or computers interconnected by a common communication medium for transmitting packetized information. To avoid collisions when different stations want to transmit information at the same time, several conflict-free protocols have been designed. One such protocol is the IEEE 802.5 token ring.

Stations on a token ring local area network are logically organized in a ring topology. Traffic on the communication medium is usually unidirectional, so that each station receives messages from one of its neighbors and passes them to its other neighbor. Messages sent from a source station to a destination station are thus relayed by all intermediate stations. A permit to transmit is controlled through the use of a token circulating around the ring. Stations that have data to transmit must first acquire the token before they can transmit them. When no station is transmitting data, the token circulates around the ring and is passed from station to station until arriving at a station that needs to transmit data. When a station needs to transmit data, it modifies the token into a busy token and inserts the data when it sends out the token. The transmitting station is responsible for removing the busy token from the ring and for generating a new free token when its transmission is over.

With respect to the time at which a new free token is generated by the transmitting station, distinction is made between the *multiple-token*, *single-token*, and *single-message* operations. For multiple-token operation, the transmitting station generates a new free token and places it immediately after the last bit of a transmitted message. Therefore, for a long ring, the chances are that there are several messages and a free token on the ring at one time. For single-token operation, a new free token is generated as soon as the transmitted message returns. If a message is long, the transmitting station will receive the busy token before it has finished transmitting. In this case, a free token is generated only after the last bit of a message has been transmitted, as in the multiple-token operation. For single-message operation, a new free token is generated only after the last bit of the transmitted message has been returned and erased. Both single-token and single-message operations ensure that there is at most one free or busy token on the ring at all times.

A token ring may be viewed as a polling system where the server represents the token

and the queues represent the stations. Customers arriving at the queues correspond to the messages that need to be transmitted by the stations. The switch-over time is the time needed for a token to pass from one station to the other and is a combination of the ring speed, station latency and the propagation time. The service times are determined according to the free-token generation policies discussed above and consist of the ring speed, message size, switch-over times and the station latency.

2.2 Flexible manufacturing and production systems

Polling models are also useful for modeling flexible manufacturing and production systems, where machines can be used to perform various types of task. Here, the server typically represents the machine, each of the queues represents a different type of job and the switch-over times represent the times needed by the machine to change from one type of operation to another. A typical application is multi-product economic lot scheduling, in which different types of product have to be processed by a single machine.

2.2.1 Multi-product economic lot scheduling

Scheduling production of multiple products on a single machine under tight capacity constraints is one of the classic problems in operations research. There exist many variations of multi-product single-machine scheduling problems, but these can mainly be classified by the following three characteristics.

1. *Presence or absence of setup times and/or costs.* The most important impact of setups on the production plan is that the products need to be produced in batches, since otherwise costly capacity is wasted on setups. Furthermore, setup times make it impossible to be completely responsive to the demand.

2. *Customized or standardized products.* Since customized products can only be produced when there is a request for an order, these products have to be produced in a make-to-order fashion. In case of standardized products one may choose a make-to-stock production policy, because such products do not have to be produced to customer specifications. It is obvious that standardized products thus give more freedom in deciding when to make which product and in what quantity.

3. *Stochastic or deterministic environment.* In a completely deterministic environment one can confine oneself to a rigid production plan which is repeated over and over again. However, when the company has to be responsive to a stochastic environment such a rigid schedule will not suffice anymore.

The so-called stochastic economic lot scheduling problem deals with the make-to-stock production of multiple standardized products on a single machine with limited capacity

under random demands, possibly random setup times and possibly random production times. This is a common problem in practice, e.g., in glass and paper production, injection molding, metal stamping and semi-continuous chemical processes, but also in bulk production of consumer products such as beers and shoes. The stochastic economic lot scheduling problem is analyzed by Winands [48].

The following class of fixed-sequence base-stock policies is used in many firms for the control of the inventory of each product. N products are distinguished, which are numbered $1, \ldots, N$. To each individual product a stock point is assigned which is controlled by a base-stock inventory policy. Under such a policy, for each product there exists a predefined desired number of items in stock, the base-stock level b_i, $i = 1, \ldots, N$. When demand arrives at a stock point and the requested product is on stock, the demand is immediately fulfilled. Otherwise, demand is backlogged and fulfilled as soon as the product becomes available after production. A production order, also called replenishment order, is placed immediately after demand for the corresponding product has arrived. These production orders queue up at the production facility, where each product has its own designated queue. The products are produced according to a fixed production sequence. When the machine starts production of a product, it will continue production until either the base-stock level has been reached or a second local criterion, i.e., only dependent on the stock level of the product currently setup, has been fulfilled.

It is easy to see that the steady-state *shortfall* L_i of product i (i.e., the number of outstanding production orders at the production facility) is independent of the base-stock levels. This implies that the performance of the production facility can be analyzed independently of these base-stock levels. Moreover, the shortfall distribution of a product at the production facility is identical to the queue length distribution of the corresponding queue in the polling system. The arrival, service and switch-over time processes in such a polling system are identical to the demand, processing and setup time processes in the stochastic economic lot scheduling problem, respectively. For given base-stock levels, the evaluation of a fixed-sequence base-stock policy is therefore equivalent to the evaluation of the corresponding polling system.

To completely determine a fixed-sequence base-stock policy, a number of decisions have to be made: the *lot-sizing decision*, the *sequencing decision* and the *base-stock decision*. The lot-sizing decision determines how many items of each product should be produced per production run. This is equivalent to the service policy of polling systems. The sequencing decision decides on the order and frequency in which products are produced. This is equivalent to the routing scheme in terms of polling systems. The base-stock decision determines the values of the optimal base-stock levels. Since the control of the production facility is independent of the base-stock levels, optimization of the base-stock levels can be done separately from the analysis of the production facility. Given the first two decisions, the shortfall distributions can be computed by analyzing the queue length distributions in the corresponding polling system. Then, using the standard notation of inventory models

with distribution function of the shortfall F_{L_i}, holding costs h_i and backlogging costs q_i, the optimal base-stock levels b_i^* follow from the classical single-order model and are given by, for $i = 1, \ldots, N$,

$$b_i^* = \min\Big\{n \in \mathbb{N}_0 | F_{L_i}(n) \geq \frac{q_i}{q_i + h_i}\Big\}. \tag{2.1}$$

2.3 Traffic signal control

A common sight in daily life are traffic intersections. Looked at closely, a traffic intersection can be modeled as a multi-queue single-server system where the road intersection and the road lanes are represented by, respectively, the server and the queues. There is a competition between the lanes to use the intersection. In order to permit an orderly usage of the intersection by the vehicles, traffic lights are used to control access to the intersection in a pre-determined fashion. Each lane has a finite capacity and vehicles arrive according to a random pattern. Since the input to a traffic intersection is a collection of outputs of upstream traffic lights, the arrival process has some correlation. In case of traffic intersections in which each lane has a fixed time period, each lane can be analyzed separately. When the traffic lights are vehicle-actuated, each lane can no longer be analyzed as a single queue and therefore one can model it as a polling system (cf., e.g., [4] and references therein).

2.3.1 Vehicle-actuated control

Vehicle-actuated control takes into account the presence of vehicles to determine the adjustment of the traffic lights. With this system, vehicles on any intersection lane are sensed by some detecting device, e.g., magnetic loops placed in the roadway. By recording the vehicles as they cross the detector and by timing the intervals between the vehicles, the traffic lights are automatically adjusted to give preference to the lane with the heaviest flow. Vehicle-actuated control can thus take into account fluctuations in traffic flow so that, e.g., in light traffic conditions delays are less.

The most commonly used vehicle-actuated control works as follows. A traffic light only turns green when vehicles are waiting in front of the traffic light. This can be detected by short loops. When a traffic light turns green, it stays green for a minimum amount of time. After this time it stays green until the loops have detected a gap time of at least a certain duration. The effective green time is bounded by a maximum time, when the traffic light, no matter what, turns red.

Vehicle-actuated control may be translated into a polling system where the server represents the intersection and the queues represent the road lanes. Arriving vehicles are represented by the customers and the service process of the customers can be compared with the departure process of the vehicles. The switch-over time from one queue to another may be viewed as the clearance time of the intersection. The sequence in which the server polls

the queues is defined by the routing scheme, which may either be static or dynamic. If the effective green time of the traffic lights lasts until no vehicles are present anymore, then this control corresponds to the exhaustive service policy. For the control scheme as discussed above no standard service policy exists, but it may be viewed as some kind of limited service where extra conditions are imposed upon.

Note that the possibility exists for multiple road lanes to have a green light phase at the same time. This is possible if those lanes are *compatible* (i.e., they can safely cross the intersection simultaneously) with each other. In this situation a queue can be viewed as a set of lanes which are all compatible. Usually, the assumption is made that the lane with the highest occupation rate then characterizes the whole set.

Chapter 3

Analysis

This chapter is devoted to the analysis of polling systems. Because the mathematics involved is pretty advanced, not all of it is discussed in great detail. It is mainly a summary of the most important techniques and results available in the literature. It starts by describing the mathematical model and by introducing some notation. This notation is very important as it is used in the remainder of this chapter.

3.1 Model description and notation

Consider a system consisting of $N \geq 1$ queues, $Q_1, \ldots, Q_N$, in which there is infinite buffer capacity for each queue. A single server visits and serves the queues in cyclic order, switching from queue to queue. Customers arriving at Q_i are called type-i customers and arrive according to a Poisson arrival process with rate λ_i. The total arrival rate is denoted by $\Lambda = \sum_{i=1}^{N} \lambda_i$. The service time of a type-i customer is a random variable B_i, with k-th moment $b_i^{(k)}$, $k = 1, 2$. The k-th moment of the service time of an arbitrary customer is denoted by $b^{(k)} = \sum_{i=1}^{N} \lambda_i b_i^{(k)} / \Lambda$, $k = 1, 2$. The load offered to Q_i is $\rho_i = \lambda_i b_i^{(1)}$, and the total offered load is equal to $\rho = \sum_{i=1}^{N} \rho_i$. Define a polling instant at Q_i as a time epoch at which the server visits Q_i. The server serves each queue according to some service policy. Upon departure from Q_i the server immediately proceeds to $Q_{(i \mod N)+1}$, incurring a switch-over time R_i, with k-th moment $r_i^{(k)}$, $k = 1, 2$. Denote by $r = \sum_{i=1}^{N} r_i^{(1)}$ the expected total switch-over time per cycle of the server along the queues, and denote the second moment by $r^{(2)} = \sum_{i=1}^{N} r_i^{(2)} + \sum_{i \neq j} r_i^{(1)} r_j^{(1)}$. All interarrival times, service times and switch-over times are assumed to be mutually independent and independent of the state of the system.

The cycle length C_i is defined as the time between two consecutive polling instants at Q_i. By the balancing argument that the amount of work arriving during a cycle should on average equal the amount of work departing during a cycle (i.e., $\rho \mathbb{E}C_i = \mathbb{E}C_i - r$), it follows that the mean cycle length is given by $\mathbb{E}C_i = r/(1-\rho)$, which is independent of

the queue involved. The visit period V_i is defined as the time the server spends servicing customers at Q_i. Since the server is working a fraction ρ_i of the time on Q_i, the mean visit period is given by $\mathbb{E}V_i = \rho_i \mathbb{E}C_i$. Subsequently, the intervisit period I_i, the time between a departure and the next arrival of the server to Q_i, is defined as $I_i = C_i - V_i$.

The main quantity of interest is the waiting time W_i incurred by an arbitrary customer at Q_i, defined as the time between the arrival of a customer and the moment at which it starts to receive service. Note that all results for the mean waiting time can readily be translated into results for the mean queue length L_i — and vice versa — via Little's law: $L_i = \lambda_i W_i$.

3.2 Service policy

In the model description there is room to choose a service policy. The service policy determines how many customers are served during the visit of the server to a queue. The assumption is made that all queues are served according to the same service policy. The following policies are most commonly found in the literature.

- *Exhaustive* service: when the server visits a queue, the customers are served until the queue is empty.

- *Gated* service: when the server visits a queue, only those customers are served who were present at the polling instant.

- *Globally gated* service: at the beginning of a cycle, indicated by a polling instant at Q_1, all customers present at $Q_1, \ldots, Q_N$ are marked. Only the marked customers are served during the coming cycle. Customers who meanwhile arrive at the queues will have to wait until being marked at the beginning of the next cycle.

- *Binomial gated* service: when the server visits Q_i, a random number of customers is served having a binomial distribution with parameters X_i and p_i, where X_i is the number of customers queued in Q_i at the polling instant, and p_i is some number, $0 < p_i \leq 1$. Note that the case $p_i = 1$ amounts to the gated policy.

- *Multi-phase gated* service: Q_i consists of $K_i \geq 1$ buffers: a phase-1 buffer, a phase-2 buffer up to a phase-K_i buffer. Arriving customers enter the phase-1 buffer. When the server arrives at Q_i, it closes the gate behind the customers residing in the phase-1 buffer. Then, all customers waiting in the phase-K_i buffer are served. Subsequently, all customers before the gate at the phase-k buffer are instantaneously forwarded to the phase-$(k+1)$ buffer, $k = 1, \ldots, K_i - 1$, and the server proceeds to the next queue.

- *k-limited* service: when visiting Q_i, the server works until either $k_i \geq 1$ customers have been served or the queue becomes empty, whichever comes first. Note that the case $k_i = \infty$ is equivalent to the exhaustive policy.
- *Bernoulli* service: when the server arrives at Q_i, finding that queue non-empty, at least one customer is served. If after the completion of the service of a customer there are still customers waiting, with probability q_i, $0 \leq q_i \leq 1$, another customer at Q_i is served. Note that the cases $q_i = 0$ and $q_i = 1$ correspond to the 1-limited and the exhaustive policies, respectively.

Numerous hybrid variants of service policies can be conceived by combining the various types of cut-off mechanisms. Some of these have been studied, such as the probabilistically-limited service (cf., [20]), fractional-exhaustive service (cf., [23]) and time-limited service policies with exponential time limits (cf., [21]) and with constant time limits (cf., [39]).

Within a queue customers are served in the order defined by the *scheduling discipline*, which most frequently is in a *first come first served* order. It is obvious that the mean waiting times are the same under any work-conserving (i.e., the server does not create or destroy work and never idles during a visit to a queue) non-preemptive scheduling discipline that does not depend on the service times of the customers. However, the complete distributions of the waiting times do depend on the scheduling discipline.

3.3 Stability

The stability of a queueing system is very important. A polling system is said to be stable if it admits a stable regime (e.g., all steady-state queue lengths are finite) with integrable cycle length. In order for the cycle length to be positive and finite, $\rho < 1$ is obvious a necessary condition, but this condition is not always sufficient. In [14] the following necessary and sufficient condition for stability is derived

$$\rho + \max_{1 \leq i \leq N} \{\lambda_i / G_i^*\} r < 1, \tag{3.1}$$

where G_i^* is the maximum expected number of customers served in Q_i during a cycle and is determined by the service policy used to serve the queue. It is assumed that $\lambda_i / G_i^* = 0$ if $G_i^* = \infty$. For the exhaustive and all gated service policies $G_i^* = \infty$, and for the k-limited service policy it is obvious that $G_i^* = k_i$. This condition also provides some insight in the long-run behavior of the polling system, since in case of heavy traffic, the order in which the queues become unstable is given.

3.4 Pseudo-conservation law

In the case of zero switch-over times, a *conservation law* holds for the total amount of work in the system. This amount is independent of the service policy and equals the

amount of work in an $M/G/1$ queue with arrival rate Λ and with service time distribution $\sum_{i=1}^{N}(\lambda_i/\Lambda)B_i(\cdot)$. By applying the celebrated Pollaczek-Khintchine formula, the following conservation law is obtained

$$\sum_{i=1}^{N}\rho_i\mathbb{E}W_i = \rho\frac{\sum_{i=1}^{N}\lambda_i b_i^{(2)}}{2(1-\rho)}. \tag{3.2}$$

In the case of non-zero switch-over times, a similar relationship holds for the weighted sum of the mean waiting times. This is then called the *pseudo-conservation law*, because the amount of work is no longer independent of the service policy. On the basis of the principle of work decomposition, Boxma and Groenendijk [1] show the following classical result

$$\sum_{i=1}^{N}\rho_i\mathbb{E}W_i = \rho\frac{\sum_{i=1}^{N}\lambda_i b_i^{(2)}}{2(1-\rho)} + \rho\frac{r^{(2)}}{2r} + \frac{r}{2(1-\rho)}\left[\rho^2 - \sum_{i=1}^{N}\rho_i^2\right] + \sum_{i=1}^{N}\mathbb{E}M_i, \tag{3.3}$$

where M_i stands for the amount of work at Q_i at an arbitrary moment at which the server departs from Q_i. The term $\mathbb{E}M_i$ is completely determined by the service policy at Q_i and is independent of (the service policy at) the other queues. This law is applicable to a large variety of polling systems since for most service policies $\mathbb{E}M_i$ can be derived in closed-form.

- Exhaustive service: since the queue is empty the moment at which the server departs from Q_i, $\mathbb{E}M_i = 0$. The pseudo-conservation law can then be simplified to

$$\sum_{i=1}^{N}\rho_i\mathbb{E}W_i = \rho\frac{\sum_{i=1}^{N}\lambda_i b_i^{(2)}}{2(1-\rho)} + \rho\frac{r^{(2)}}{2r} + \frac{r}{2(1-\rho)}\left[\rho^2 - \sum_{i=1}^{N}\rho_i^2\right]. \tag{3.4}$$

- Gated service: the amount of work that remains is equal to the amount of work that arrived during the visit period of Q_i, $\mathbb{E}M_i = \rho_i^2 r/(1-\rho)$. The pseudo-conservation law can then be simplified to

$$\sum_{i=1}^{N}\rho_i\mathbb{E}W_i = \rho\frac{\sum_{i=1}^{N}\lambda_i b_i^{(2)}}{2(1-\rho)} + \rho\frac{r^{(2)}}{2r} + \frac{r}{2(1-\rho)}\left[\rho^2 + \sum_{i=1}^{N}\rho_i^2\right]. \tag{3.5}$$

- Globally gated service: Boxma et al. [2] derive the following result for the expected amount of work left at Q_i when the server leaves this queue

$$\mathbb{E}M_i = \rho_i\sum_{j=1}^{i-1}\left(\rho_j\frac{r}{1-\rho} + r_j^{(1)}\right) + \rho_i^2\frac{r}{1-\rho}. \tag{3.6}$$

The pseudo-conservation law can then be simplified to

$$\sum_{i=1}^{N} \rho_i \mathbb{E}W_i = \rho \frac{\sum_{i=1}^{N} \lambda_i b_i^{(2)}}{2(1-\rho)} + \rho \frac{r^{(2)}}{2r} + \frac{r}{1-\rho}\rho^2 + \sum_{i=2}^{N} \rho_i \sum_{j=1}^{i-1} r_j^{(1)}. \tag{3.7}$$

- Binomial gated service: Van der Mei [31] derives the following result for the binomial gated policy as a special case of cyclic polling models with general branching-type service policies

$$\mathbb{E}M_i = \frac{\rho_i(1-p_i(1-\rho_i))}{p_i} \frac{r}{1-\rho}. \tag{3.8}$$

- Multi-phase gated service: Van der Mei and Roubos [33] show the following result

$$\mathbb{E}M_i = \rho_i \left((K_i - 1) + \rho_i\right) \frac{r}{1-\rho}. \tag{3.9}$$

- k-limited service: the general form of the pseudo-conservation law as shown in (3.3) still holds. However, the unknowns $\mathbb{E}M_i$ cannot be derived in closed-form. Everitt [12] expresses these unknown quantities in terms of $g_i^{(2)}$, the second factorial moment of the number of customers served during a visit to Q_i. The following pseudo-conservation law is obtained

$$\begin{aligned}\sum_{i=1}^{N} \rho_i \left(1 - \frac{\lambda_i r}{k_i(1-\rho)}\right) \mathbb{E}W_i &= \rho \frac{\sum_{i=1}^{N} \lambda_i b_i^{(2)}}{2(1-\rho)} + \rho \frac{r^{(2)}}{2r} \\ &+ \frac{r}{2(1-\rho)} \left[\rho^2 - \sum_{i=1}^{N} \rho_i^2\right] + \frac{r}{1-\rho} \sum_{i=1}^{N} \frac{\rho_i^2}{k_i} - \sum_{i=1}^{N} \frac{\rho_i(1-\rho_i) g_i^{(2)}}{2\lambda_i k_i}.\end{aligned} \tag{3.10}$$

  For $k_i = 1$, $g_i^{(2)} = 0$, but for $k_i \neq 1$, $g_i^{(2)}$ is not known exactly.

Pseudo-conservation laws seem to be very useful in several respects. Firstly, they are useful for obtaining or testing approximations for individual mean waiting times. Such approximations are badly needed in analytically intractable cases (e.g., in the case of limited service) but also in analytically tractable cases. The latter because, when the number of queues is large, the numerical computation of the exact formulas can become very cumbersome. Secondly, pseudo-conservation laws can also be used to study asymptotics, yielding information about what happens when the number of queues becomes very large or when the offered load at a particular queue approaches its stability limit. Furthermore, it gives a relatively simple expression for the weighted sum of the mean waiting times, which may be used as a first indication of overall system performance.

3.5 Analysis techniques

There exists a striking difference in the complexity of the analysis of polling systems. If the service policy satisfies a certain property, the corresponding polling system allows an exact detailed analysis by rather standard methods. However if this property is violated, detailed exact results are very scarce and are mainly restricted to two-queue models and symmetric systems. The class of service policies which are easy to analyze satisfy the following property [37].

Property 3.1. If the server arrives at Q_i to find k_i customers there, then during the course of the server's visit, each of these k_i customers will effectively be replaced in an i.i.d. manner by a random population having probability generating function $h_i(\underline{z}) = h_i(z_1, \ldots, z_N)$, which can be any N-dimensional probability generating function.

For instance, the gated service policy satisfies Property 3.1, because at the end of the server's visit, each customer present at the beginning of the visit has effectively been replaced in an i.i.d. manner by all customers who have arrived during its service. Similarly, it is readily verified that, e.g., the exhaustive policy, the binomial gated policy and the fractional-exhaustive policy satisfy Property 3.1. Property 3.1 does not hold for the k-limited service policy. At the end of the server's visit to Q_i, each of the served customers has effectively been replaced by a population of all customers who arrived during its service. The other customers present at a polling instant are not served at all and are each "replaced" by a single customer at the queue under consideration. Consequently, not all customers are replaced in an i.i.d. manner and Property 3.1 is violated. Other service policies violating Property 3.1 are, e.g., the Bernoulli policy and the time-limited policy.

Example 3.1. The k-limited service policy does not satisfy Property 3.1 and is therefore very hard to analyze. However, in case of a symmetric system some performance measures can easily be obtained. To show this, consider the model defined by the following parameters. There are $N = 3$ queues, arrivals occur according to a Poisson arrival process with rate 0.25 and the service times are gamma distributed with shape parameter 0.5 and scale parameter 2 for each queue. The server serves the queues according to the 1-limited service policy and the switch-over times are uniformly distributed over the interval $[0.2, 0.4]$.

A gamma distributed random variable with shape parameter k and scale parameter θ has mean $k\theta$, variance $k\theta^2$ and (hence) second moment $(k + k^2)\theta^2$. So $b_i^{(1)} = 1$ and $b_i^{(2)} = 3$, $i = 1, 2, 3$. Since $\lambda_i = 0.25$, $\rho_i = 0.25$, $i = 1, 2, 3$, $\Lambda = 0.75$ and $\rho = 0.75$. The total switch-over time is $r = 0.9$ and the second moment is $r^{(2)} = 0.82$. Note that this system is stable since the necessary and sufficient condition for stability (3.1) holds: $0.975 < 1$. Because the system is symmetric, the waiting times at the queues are equal. Denote this waiting time by W. The mean waiting time can now directly be computed using the pseudo-conservation law in (3.10). It follows that $\mathbb{E}W = 608/9 \approx 67.56$ and then that $\mathbb{E}L = 152/9 \approx 16.89$.

Although the class of service policies which satisfy Property 3.1 allows, at least formally, an exact analysis, the problem of efficiently determining numerical values for performance measures like mean waiting times and mean queue lengths is generally non-trivial. In the past several numerical approaches have been proposed for computing these performance measures. In the next subsections an overview is given of some of the available techniques.

3.5.1 Buffer occupancy method

One such technique is the *buffer occupancy* method as developed in [7, 8, 11]. This method is based on the buffer occupancy variables $X_{i,j}$, which denote the queue length at queue j at a polling instant at queue i, $i, j = 1, \ldots, N$. The relationship between queue i and queue $i+1$ is used to obtain expressions for the mean queue length, $\mathbb{E}X_{i,j}$. For the exhaustive service policy these relations are given by

$$f_{i+1}(i) = r_i^{(1)}\lambda_i, \tag{3.11a}$$

$$f_{i+1}(j) = r_i^{(1)}\lambda_j + f_i(j) + f_i(i)\frac{\lambda_j b_i^{(1)}}{1-\rho_i}, \quad j \neq i. \tag{3.11b}$$

The solution is given by

$$\mathbb{E}X_{i,i} = f_i(i) = \lambda_i(1-\rho_i)\frac{r}{1-\rho}, \tag{3.12a}$$

$$\mathbb{E}X_{i,j} = f_i(j) = \lambda_j\left(\sum_{k=j}^{i-1} r_k^{(1)} + \frac{r\sum_{k=j+1}^{i-1}\rho_k}{1-\rho}\right), \quad j \neq i. \tag{3.12b}$$

For the gated service policy these relations are given by

$$f_{i+1}(i) = r_i^{(1)}\lambda_i + \rho_i f_i(i), \tag{3.13a}$$

$$f_{i+1}(j) = r_i^{(1)}\lambda_j + \lambda_j b_i^{(1)} f_i(i) + f_i(j), \quad j \neq i. \tag{3.13b}$$

The solution is given by

$$\mathbb{E}X_{i,i} = f_i(i) = \lambda_i\frac{r}{1-\rho}, \tag{3.14a}$$

$$\mathbb{E}X_{i,j} = f_i(j) = \lambda_j\left(\sum_{k=j}^{i-1} r_k^{(1)} + \frac{r\sum_{k=j}^{i-1}\rho_k}{1-\rho}\right), \quad j \neq i. \tag{3.14b}$$

The buffer occupancy method requires the solution of N^3 linear equations with unknowns $\mathbb{E}X_{i,j}X_{j,k}$ to compute the mean waiting times in all N stations simultaneously. These equations may be efficiently solved in an iterative manner requiring $O(N^3 \log_\rho \epsilon)$ elementary operations, where ρ is the total occupation rate and ϵ is the relative accuracy required [24]. The buffer occupancy method is applicable to the complete class of service policies satisfying Property 3.1. However, it is limited to systems in which the interarrival time is exponentially distributed.

3.5.2 Descendant set approach

Based on this buffer occupancy method, the *descendant set* approach is developed [17]; an iterative technique that computes the mean waiting time at each queue independently of the other queues. The descendant set approach is based on counting the number of descendants of each customer in the system. Each customer is classified as either an original customer or a non-original customer. An original customer is a customer who arrives at the system during a switch-over period and a non-original customer is a customer who arrives at the system during the service of another customer. The queue length distribution at polling instants is then derived based on the relationship between the number of original and non-original customers. This method requires $O(N \log_\rho \epsilon)$ elementary operations for the computation of the mean waiting time in a single station. These mean waiting times are given by, for the exhaustive service policy,

$$\begin{aligned}\mathbb{E}W_k = \frac{1+\rho_k}{2}\frac{r}{1-\rho} + \frac{1}{2(1-\rho_k)}\Bigg[\lambda_k b_k^{(2)} + \left(r_{k-1}^{(2)} - \left(r_{k-1}^{(1)}\right)^2\right)\frac{1-\rho}{r} \\ + \frac{1}{\lambda_k^2}\sum_{i=1}^{N}\frac{\psi_{i,k}}{\rho_i^2}\left(\lambda_i b_i^{(2)} + \left(r_{i-1}^{(2)} - \left(r_{i-1}^{(1)}\right)^2\right)\frac{1-\rho}{r}\right)\Bigg],\end{aligned} \tag{3.15}$$

and for the gated service policy by

$$\mathbb{E}W_k = \frac{1+\rho_k}{2}\frac{r}{1-\rho} + \frac{1+\rho_k}{2\lambda_k^2}\sum_{i=1}^{N}\frac{\psi_{i,k}}{\rho_i^2}\left(\lambda_i b_i^{(2)} + \left(r_i^{(2)} - \left(r_i^{(1)}\right)^2\right)\frac{1-\rho}{r}\right), \tag{3.16}$$

where

$$\psi_{i,k} = \begin{cases} \lambda_i^2 \displaystyle\sum_{c=1}^{\infty} \alpha_{(i,c),k}^2, & i = 1, \ldots, k-1, \\ \lambda_i^2 \displaystyle\sum_{c=1}^{\infty} \alpha_{(i,c-1),k}^2, & i = k, \ldots, N. \end{cases} \tag{3.17}$$

The coefficients $\alpha_{(i,c),k}$ are given by, for the exhaustive service policy, for $c = 0, 1, \ldots,$

$$\alpha_{(i,c),k} = \frac{b_i^{(1)}}{1-\rho_i}\left[\sum_{j=i+1}^{N}\lambda_j \alpha_{(j,c),k} + \sum_{j=1}^{i-1}\lambda_j \alpha_{(j,c-1),k}\right], \tag{3.18}$$

and for the gated service policy by, for $c = 0, 1, \ldots,$

$$\alpha_{(i,c),k} = b_i^{(1)}\left[\sum_{j=i+1}^{N}\lambda_j \alpha_{(j,c),k} + \sum_{j=1}^{i}\lambda_j \alpha_{(j,c-1),k}\right]. \tag{3.19}$$

Starting with the initial values $\alpha_{(k,0),k} = 1$, $\alpha_{(i,0),k} = 0$, $i = k+1, \ldots, N$, $\alpha_{(i,-1),k} = 0$, $i = 1, \ldots, k-1$, all coefficients $\alpha_{(i,c),k}$ can then be recursively determined.

Like the buffer occupancy method, the descendant set approach can be applied to all variations of polling systems with the service policy satisfying Property 3.1 in which customers arrive according to a Poisson arrival process. The main advantage of the descendant set approach is that the waiting time computation of one queue is independent of that of the others, so that its superiority is mostly expressed when not all N waiting times are needed.

3.5.3 Individual station technique

A second well-known method based on the buffer occupancy method is the *individual station* technique [40], which also allows, as the name suggests, the individual computation of the mean waiting time at each queue. The individual station technique is, however, not an iterative approach. The mean waiting time at a single queue is computed in $O(N^2)$ elementary operations, which obviously does not depend on the system utilization contrary to the computational complexities of the aforementioned methods. A characteristic of the individual station technique is that the accuracy typically does not degrade significantly when the system is heavily loaded, while this is the case with the descendant set approach. Therefore, the descendant set approach and the individual station technique can be considered as complementary to each other. This technique is also applicable to the same class of polling systems as the buffer occupancy method and the descendant set approach.

3.5.4 Station time technique

Besides the techniques based on the buffer occupancy method, a completely different technique for solving the mean waiting times exists based on the *station time* [13]. The station time technique computes all mean waiting times simultaneously starting from the station time variables S_i, $i = 1, \ldots, N$. The station time S_i is composed of the time the server spends servicing customers at queue i plus the preceding switch-over time in case of exhaustive service or plus the succeeding switch-over time in case of gated service. The station time technique induces a set of N^2 linear equations with unknowns $\mathbb{E}S_iS_j$, which can be solved iteratively in $O(N^2 \log_\rho \epsilon)$ elementary operations leading to all N mean waiting times. The station time technique can be applied to a restricted class of polling systems only, that is, to polling systems with either exhaustive or gated service at all queues.

An extension of the station time technique is the approach developed in [38] and induces a set of only N linear equations. However, the resulting N equations are less sparse and the benefit of reducing the number of equations is off set by using a numerical method that requires $O(N^3)$ elementary operations to obtain all N mean waiting times.

3.5.5 Approximate methods

One method to approximate the behavior of polling systems is based on the decomposition approach (cf., e.g., [6, 19, 22, 49]). In this method each queue is treated separately as a single-server queue with vacation. The analysis is done in two stages. In the first part, which is exact, the performance measures of the single-server queue with vacation are derived. The second part of the analysis focuses on obtaining an approximation for the vacation period distribution. When possible, the vacation period distribution is taken as the convolution of the visit periods of the other $N-1$ queues. However, when the vacation period does not lend itself to a simple convolution of the visit periods, an approximation of the vacation period based on a dependent and an independent part is taken. In either case, using an iterative procedure the decomposition approach converges fairly fast to within an acceptable error. This approach is being used more frequently because more realistic traffic models no longer assume Poisson arrivals, but rather bursty traffic, e.g., packetized voice and data traffic. Also, the limited service policy is emerging as the preferred service policy, as reflected by several ANSI/IEEE standards such as the IEEE 802.4 token-passing bus method.

Another method that can be seen as an approximation is based on heavy-traffic asymptotics (cf., e.g., [29, 30, 32]). Heavy-traffic asymptotics give closed-form expressions for the complete waiting time distributions when the load tends to unity ($\rho \uparrow 1$). The analysis of heavy-traffic asymptotics has been motivated by the observations that the efficiency of numerical techniques degrade significantly for heavily loaded systems and that exact closed-form expressions provide much more insight into the dependence of the performance measures on the system parameters. The results in the limiting case can easily be used for approximating the distributions of the waiting times for stable systems, using a linear scaling in the offered load. The time required to evaluate these approximations is negligible, but the downside is that they are only accurate for high values of the load, typically in the range 85–90% or more. Heavy-traffic asymptotics can be applied to all polling systems satisfying Property 3.1.

3.5.6 Results

In this subsection the most important results of the analysis for the exhaustive and gated service policies are presented. The complete algorithms for computing the mean waiting times are given together with an example to illustrate how these algorithms can be applied.

Exhaustive service

The most efficient algorithm for computing the mean waiting times in case of the exhaustive service policy is given by

$$\mathbb{E}W_i = \frac{\mathbb{E}I_i^2}{2\mathbb{E}I_i} + \frac{\lambda_i b_i^{(2)}}{2(1-\rho_i)}, \tag{3.20}$$

where

$$\mathbb{E}I_i = \frac{(1-\rho_i)r}{1-\rho} \tag{3.21}$$

and

$$\mathbb{E}I_i^2 = r_{i-1}^{(2)} - \left(r_{i-1}^{(1)}\right)^2 + \frac{1-\rho_i}{\rho_i}\sum_{\substack{j=1\\ j\neq i}}^{N} r_{ij} + (\mathbb{E}I_i)^2. \tag{3.22}$$

Here, r_{ij} represents the covariance of the station time for queues i and j, where for exhaustive service systems, the station time for queue i is defined as the time interval between the successive instants when the server leaves queue $i-1$ and queue i. The set $\{r_{ij}; i,j = 1,\ldots,N\}$ is computed by solving a system of N^2 linear equations

$$r_{ij} = \frac{\rho_i}{1-\rho_i}\left(\sum_{m=i+1}^{N} r_{jm} + \sum_{m=1}^{j-1} r_{jm} + \sum_{m=j}^{i-1} r_{mj}\right), \quad j < i, \tag{3.23a}$$

$$r_{ij} = \frac{\rho_i}{1-\rho_i}\left(\sum_{m=i+1}^{j-1} r_{jm} + \sum_{m=j}^{N} r_{mj} + \sum_{m=1}^{i-1} r_{mj}\right), \quad j > i, \tag{3.23b}$$

$$r_{ii} = \frac{r_{i-1}^{(2)} - \left(r_{i-1}^{(1)}\right)^2}{(1-\rho_i)^2} + \frac{\lambda_i b_i^{(2)}\mathbb{E}I_i}{(1-\rho_i)^3} + \frac{\rho_i}{1-\rho_i}\sum_{\substack{j=1\\ j\neq i}}^{N} r_{ij}. \tag{3.23c}$$

Example 3.2. Consider a three-queue polling system with exhaustive service. Arrivals occur according to Poisson arrival processes with rates 0.3, 0.4 and 0.2 for queue 1, 2 and 3, respectively. The service times at queue 1, 2 and 3 are exponentially distributed with mean 1, uniformly distributed over the interval $[0,1]$, and gamma distributed with shape parameter 1 and scale parameter 2, respectively. The switch-over times from queue 1 to 2 are exponentially distributed with rate 1, those from queue 2 to 3 are exponentially distributed with rate 2, and those from queue 3 to 1 are exponentially distributed with rate 3. The following system parameters can then be identified

$$\begin{array}{lllllll}
\lambda_1 = \frac{3}{10}, & b_1^{(1)} = 1, & b_1^{(2)} = 2, & \rho_1 = \frac{3}{10}, & r_1^{(1)} = 1, & r_1^{(2)} = 2,\\
\lambda_2 = \frac{2}{5}, & b_2^{(1)} = \frac{1}{2}, & b_2^{(2)} = \frac{1}{3}, & \rho_2 = \frac{1}{5}, & r_2^{(1)} = \frac{1}{2}, & r_2^{(2)} = \frac{1}{2},\\
\lambda_3 = \frac{1}{5}, & b_3^{(1)} = 2, & b_3^{(2)} = 8, & \rho_3 = \frac{2}{5}, & r_3^{(1)} = \frac{1}{3}, & r_3^{(2)} = \frac{2}{9},\\
\rho = \frac{9}{10}, & r = \frac{11}{6}, & \mathbb{E}C = \frac{55}{3}, & \mathbb{E}I_1 = \frac{77}{6}, & \mathbb{E}I_2 = \frac{44}{3}, & \mathbb{E}I_3 = 11.
\end{array}$$

Further, the following set of linear equations is obtained for computing r_{ij}

$$\begin{aligned}
r_{11} &= \frac{10000}{441} + \frac{3}{7}r_{12} + \frac{3}{7}r_{13},\\
r_{22} &= \frac{775}{144} + \frac{1}{4}r_{21} + \frac{1}{4}r_{23},\\
r_{33} &= \frac{8875}{108} + \frac{2}{3}r_{31} + \frac{2}{3}r_{32},\\
r_{12} &= \frac{3}{7}r_{22} + \frac{3}{7}r_{32},\\
r_{13} &= \frac{3}{7}r_{32} + \frac{3}{7}r_{33},\\
r_{21} &= \frac{1}{4}r_{13} + \frac{1}{4}r_{11},\\
r_{23} &= \frac{1}{4}r_{33} + \frac{1}{4}r_{13},\\
r_{31} &= \frac{2}{3}r_{11} + \frac{2}{3}r_{21},\\
r_{32} &= \frac{2}{3}r_{21} + \frac{2}{3}r_{22},
\end{aligned}$$

with solution

$$\begin{aligned}
r_{11} &= \frac{370555}{4716}, & r_{12} &= \frac{1771415}{49518}, & r_{13} &= \frac{1339195}{14148},\\
r_{21} &= \frac{612715}{14148}, & r_{22} &= \frac{115870}{3537}, & r_{23} &= \frac{17345}{262},\\
r_{31} &= \frac{862190}{10611}, & r_{32} &= \frac{1076195}{21222}, & r_{33} &= \frac{2407325}{14148}.
\end{aligned}$$

From these, the second moments of the intervisit periods can be computed to obtain

$$\mathbb{E}I_1^2 = \frac{9956101}{21222}, \qquad \mathbb{E}I_2^2 = \frac{2313730}{3537}, \qquad \mathbb{E}I_3^2 = \frac{41815}{131}.$$

Finally, the mean waiting times can now easily be computed. One obtains

$$\mathbb{E}W_1 = \frac{1455649}{77814} \approx 18.71,\ \mathbb{E}W_2 = \frac{290297}{12969} \approx 22.38,\ \mathbb{E}W_3 = \frac{136973}{8646} \approx 15.84.$$

Gated service

The mean waiting times in gated service systems are given by

$$\mathbb{E}W_i = \frac{(1+\rho_i)\mathbb{E}C_i^2}{2\mathbb{E}C}, \tag{3.24}$$

where

$$\mathbb{E}C = \frac{r}{1-\rho} \tag{3.25}$$

and

$$\mathbb{E}C_i^2 = \frac{1}{\rho_i}\sum_{\substack{j=1\\ j\neq i}}^{N} r_{ij} + \sum_{j=1}^{N} r_{ji} + (\mathbb{E}C)^2. \tag{3.26}$$

Here, r_{ij} is again the covariance of the station time for queues i and j, but the station time for queue i for gated service is defined as the time interval between the successive instants when the server visits queue i and queue $i+1$. The set $\{r_{ij}; i,j = 1,\ldots,N\}$ is given as a solution to the following set of N^2 linear equations

$$r_{ij} = \rho_i \left(\sum_{m=i}^{N} r_{jm} + \sum_{m=1}^{j-1} r_{jm} + \sum_{m=j}^{i-1} r_{mj} \right), \quad j < i, \tag{3.27a}$$

$$r_{ij} = \rho_i \left(\sum_{m=i}^{j-1} r_{jm} + \sum_{m=j}^{N} r_{mj} + \sum_{m=1}^{i-1} r_{mj} \right), \quad j > i, \tag{3.27b}$$

$$r_{ii} = r_i^{(2)} - \left(r_i^{(1)}\right)^2 + \lambda_i b_i^{(2)} \mathbb{E}C + \rho_i \sum_{\substack{j=1\\ j\neq i}}^{N} r_{ij} + \rho_i^2 \sum_{j=1}^{N} r_{ji}. \tag{3.27c}$$

Example 3.3. All input parameters are taken the same as in Example 3.2, but the customers are now served according to the gated service policy. The following set of linear equations can be obtained

$$\begin{aligned}
r_{11} &= 12 + \frac{3}{10}r_{12} + \frac{3}{10}r_{13} + \frac{9}{100}r_{11} + \frac{9}{100}r_{21} + \frac{9}{100}r_{31},\\
r_{22} &= \frac{97}{36} + \frac{1}{5}r_{21} + \frac{1}{5}r_{23} + \frac{1}{25}r_{12} + \frac{1}{25}r_{22} + \frac{1}{25}r_{32},\\
r_{33} &= \frac{265}{9} + \frac{2}{5}r_{31} + \frac{2}{5}r_{32} + \frac{4}{25}r_{13} + \frac{4}{25}r_{23} + \frac{4}{25}r_{33},\\
r_{12} &= \frac{3}{10}r_{21} + \frac{3}{10}r_{22} + \frac{3}{10}r_{32},\\
r_{13} &= \frac{3}{10}r_{31} + \frac{3}{10}r_{32} + \frac{3}{10}r_{33},\\
r_{21} &= \frac{1}{5}r_{12} + \frac{1}{5}r_{13} + \frac{1}{5}r_{11},\\
r_{23} &= \frac{1}{5}r_{32} + \frac{1}{5}r_{33} + \frac{1}{5}r_{13},
\end{aligned}$$

$$r_{31} = \frac{2}{5}r_{13} + \frac{2}{5}r_{11} + \frac{2}{5}r_{21},$$
$$r_{32} = \frac{2}{5}r_{23} + \frac{2}{5}r_{21} + \frac{2}{5}r_{22}.$$

The solution of this set reads

$$r_{11} = \frac{469051705}{11319484}, \qquad r_{12} = \frac{165387910}{8489613}, \qquad r_{13} = \frac{263822095}{5659742},$$
$$r_{21} = \frac{730327865}{33958452}, \qquad r_{22} = \frac{1613606705}{101875356}, \qquad r_{23} = \frac{179040275}{5659742},$$
$$r_{31} = \frac{372041555}{8489613}, \qquad r_{32} = \frac{702731525}{25468839}, \qquad r_{33} = \frac{2138475235}{25468839}.$$

The second moments of the cycle lengths are then given by

$$\mathbb{E}C_1^2 = \frac{16890807200}{25468839}, \qquad \mathbb{E}C_2^2 = \frac{33859584775}{50937678}, \qquad \mathbb{E}C_3^2 = \frac{17238856150}{25468839}.$$

Finally, the mean waiting times can be quantified

$$\mathbb{E}W_1 = \frac{2195804936}{93385743} \approx 23.51, \qquad \mathbb{E}W_2 = \frac{1354383391}{62257162} \approx 21.75,$$
$$\mathbb{E}W_3 = \frac{2413439861}{93385743} \approx 25.84.$$

When comparing these waiting times to the values obtained in Example 3.2, two observations can be made. Firstly, the waiting times in queues 1 and 3 have increased, whereas the waiting time in queue 2 has become smaller. Secondly, a well-known qualitative property of polling systems comes to light, i.e., in exhaustive systems heavily loaded queues experience lower waiting times than lightly loaded queues, whereas in gated systems the opposite is true.

3.6 General parameter settings

Many queueing models require that the interarrival times are independent and exponentially distributed. The results of the analysis presented in this chapter rely on these assumptions as well. However, in a wide range of applications assuming exponential interarrival times is not appropriate, whereas the independence assumptions do appear to be valid. Therefore, in the more general case, one can model the arriving customers at all queues as independent general renewal processes. The mean and second moment of the interarrival times are denoted by $\mathbb{E}A_i$ and $\mathbb{E}A_i^2$, $i = 1, \ldots, N$, respectively. The arrival rate at Q_i is then denoted by $\lambda_i = 1/\mathbb{E}A_i$. So far, hardly any exact results have been derived for polling systems with general arrival processes apart from stability conditions and some mean value results for global performance measures such as cycle lengths.

The same situation is to some extend applicable to the service times. In some transportation systems, service times at the beginning of a phase tend to be longer than the rest; they are not identically distributed. It has been observed that during the green phase of a traffic signal, the first two or three headways (i.e., times between cars in the same lane) are longer on average than the rest. Something similar happens for boarding and alighting passengers in buses and elevators.

Concerning the routing scheme most results are available on cyclic routing, but there are many systems in which the server does not visit all the queues exactly in cyclic order. For example, the physical structure of the system may require the server to visit queues first in one direction and then in the reverse direction. Such cases apply to the elevator in a building and to the scanning policy in the moving-arm disk device of a computer. Systems may be designed so as to visit some queues more often than others in a cycle to establish priority service, hence the rise of periodic routing according to a polling table. Some results are available for these systems in which the server movement does not depend on the state of the system.

Chapter 4

Simulation

Simulation is a widely used technique for computing performance measures of all kinds of models, such as queueing models. However, in spite of their enormous flexibility, simulation techniques may be rather inefficient in many cases. For instance, when in a polling model the switch-over times are small, the majority of events will be switch-over completion epochs. This is because the server will be quickly spinning around in the system when the server is empty for some time interval. Moreover, in many cases the results based on simulation are relatively inaccurate compared with numerical algorithms. Nevertheless, simulation is in many polling systems the only possibility for obtaining these performance measures because, e.g., the service policy violates Property 3.1 and therefore does not allow an exact analysis. In addition, derivatives of the system performance measures can be obtained in one simulation run, opening many possibilities for optimization purposes.

4.1 Simulation program

To simulate a polling system, a discrete-event simulation program has been implemented in the programming language Java. This simulation program is able to simulate a wide diversity of polling systems by varying one of its many parameters. These parameters include the following.

Simulation time. The simulation program continues simulating until a predefined maximum system time has been reached. During this time period arrivals, departures and server switches occur and various counters are incremented accordingly to be able to output the desired performance measures. Alternatively, one can choose to omit inputting a simulation time and to select the option to continue simulating until the performance measures have been converged. In this case, first a minimum amount of time is simulated and after that the time is gradually increased until the mean waiting times do not change more than a small fixed percentage.

Warming-up time. When the simulation starts, the system resides in an empty state. This state is usually not representative for the state of the system in equilibrium. Therefore, it is wise to simulate a certain time such that the effects of an empty system have disappeared, before beginning with the real simulation. The counters that may have increased during the warming-up period are reinitialized in order not to influence the start of the real simulation.

Number of queues. This parameter specifies how many queues the server has to visit. After one has entered the desired number of queues, the possibility of choosing the arrival process, service process and switch-over process for each queue becomes available.

Arrival process. The arrival process is characterized by the interarrival time distribution. This is the distribution of time between two subsequent customer arrivals to the same queue. The traditional Poisson arrival process with rate λ corresponds to an interarrival time that has an exponential distribution with parameter λ. One can choose from an exponential distribution, a gamma distribution, a deterministic distribution, a two-phase hyper-exponential distribution, a log-normal distribution and a uniform distribution.

Service process. The service process determines the type of probability distribution of time it takes for the server to finish servicing a customer. One can choose from an exponential distribution, a gamma distribution, a deterministic distribution, a two-phase hyper-exponential distribution, a log-normal distribution and a uniform distribution.

Switch-over process. After the server is finished servicing customers in a queue, it switches to the next queue. In the simulation program switching takes some time that is only dependent on the queue the server switches from. The switch-over process is defined by another probability distribution. One can choose from an exponential distribution, a gamma distribution, a deterministic distribution, a two-phase hyper-exponential distribution, a log-normal distribution and a uniform distribution. The simulation program is able to handle special systems in which the switch-over times are zero. An option is present that puts all switch-over processes at a deterministic probability distribution with value zero to achieve this.

Service policy. The number of customers that are served during the visit of the server to a queue is determined by the service policy. One can choose from the exhaustive, gated, globally gated, binomial gated, multi-phase gated and k-limited service policies. The last three service policies require the input of an extra parameter for each queue, which is made available as soon as one of those is selected. The binomial gated

service policy requires a probability, while the multi-phase gated and k-limited service policies require a natural number. All queues are served according to the selected service policy, but the parameters, if present, may be different.

Routing scheme. When the server is finished servicing customers in a queue, as defined by the service policy, it switches to the next queue. The order in which the queues are visited and served by the server is defined by the routing scheme. The simulation program allows the server to visit the queues periodically according to a polling table. This polling table may be filled in to anyone's liking. Default presets are available for the traditional cyclic routing, star routing and elevator routing. Additionally, a second kind of static routing is offered: *Markovian* routing, where the server switches from Q_i to Q_j with probability p_{ij}, $i, j = 1, \ldots, N$, $\sum_{j=1}^{N} p_{ij} = 1$, $i = 1, \ldots, N$.

Customer routing. Customers who have been served by the server normally leave the system. The simulation program allows routing of customers, where on completion of service at Q_i the customer goes to Q_j with probability p_{ij} and the customer leaves the system with probability p_{i0}, $i, j = 1, \ldots, N$, $\sum_{j=0}^{N} p_{ij} = 1$, $i = 1, \ldots, N$.

The simulation program uses all these parameters for a long-term simulation of the behavior of the system. After the simulation run is completed, several performance measures have been computed. These performance measures consist of, for each queue, the means and confidence intervals of the waiting time and the queue length. The simulation program is also able to compute higher moments and tail probabilities of the waiting times. Which moments and tail probabilities have to be computed, can be defined before the simulation begins. A simple strategy to determine which tail probabilities are useful is to simulate for a short time in order to get an indication of the order of magnitude of the waiting times. This should only take about a second in real-time. Useful tail probabilities are then values in the range between zero and twice the obtained mean waiting times. The confidence intervals of these higher moments and tail probabilities are included in the output as well.

The output of the simulation program is divided into three parts. The first part lists the inputted parameters, together with the total offered load to the system. This is to be able to easily distinguish the output of one simulation run from others. The second part reports the performance measures, as discussed above. These are the means and confidence intervals of the waiting times and the queue lengths, and those of the higher moments and tail probabilities of the waiting times. The third part of the output is concerned with verification. In here, the pseudo-conservation law (i.e., the weighted sum of the mean waiting times) is computed both analytically and by using the obtained results of the simulation. The relative difference between these two values gives an indication of the precision of the simulation. The verification part is only present if the pseudo-conservation law can be computed analytically, which is in all systems with Poisson arrivals, a cyclic routing scheme and no customer routing, except for the k-limited service policy where $k_i \neq 1$. In this case the confidence intervals of the mean waiting times can be used as an indication of the accuracy.

The simulation program makes use of long-term simulation in order to compute several performance measures, of which the mean and confidence interval are presented in the output. During the simulation a large amount of observations for a certain performance measure is obtained. These observations can be used straightforwardly for a point estimation, but a complicating factor arises when constructing the confidence interval: the observations are strongly correlated. In order to get around this problem the *batch-means method* is used. The observations are aggregated in such a way that the aggregated observations are approximately independent of each other. This can be achieved by dividing the simulation run into a number of subruns, each of them having a sufficiently large amount of observations. The aggregated observations are then formed by taking the mean of those observations in a subrun. It does not matter if the original observations or the aggregated observations are used for computing the point estimate. However, if the number of observations in a subrun is sufficiently large, then the aggregated observations are approximately independent of each other. In addition, if the number of observations in a subrun is sufficiently large, then the aggregated observations can be for practical purposes assumed to be normally distributed. The confidence interval can now be constructed in the traditional way using as percentile that of the Student-t distribution with the number of subruns minus one degrees of freedom.

In addition to the previously mentioned properties, the simulation program offers the following features.

1. *The ability to save and load settings.* All parameter settings can be saved to a file and can any time later be loaded again to restore all input. This is mainly convenient if one needs to run several experiments, and each experiment is just slightly different from the others.

2. *Error checking.* Before the simulation starts all parameters are checked to make sure that all entries are valid, that there are no missing values and that the polling system is stable. The simulation starts if everything is all right. Otherwise, a warning message is displayed with a clear message indicating the mistake. In addition to the error checking, some fields like, e.g., the parameters of a probability distribution, allow only valid entries and are being checked while typing.

3. *Graphical user interface and command line versions.* The simulation program features a complete graphical user interface, but can also be used through the command line. When working with the command line, simulation starts by specifying the file containing the parameter settings. The command line version can typically be used for simulation of a batch of experiments and when one is interested in a function of the waiting times.

4. *Multi-threaded.* When the simulation program is busy running a simulation, the user interface maintains its fully responsiveness to user input. It also allows running

multiple simulations at the same time, however each experiment incurs a reduction in speed then.

5. *Easy to extend.* Because the simulation program is implemented in the programming language Java and because it is divided into small components, it can easily be extended. To give an example, probability distributions can easily be added to the existing ones provided that a method to generate random variables exists. The only requirements that a probability distribution has to meet are that the mean and second moment are finite and that it can take only positive values.

4.2 Accuracy of simulation

The results based on simulation are often relatively inaccurate compared with numerical algorithms. To obtain reliable results with simulation, one has to simulate for a sufficiently long period. How long is sufficient depends on many factors such as the offered load with respect to the system's stability limit and the variability of the arrival processes, service processes and switch-over processes. The service policy may also influence the accuracy of simulation. The following example demonstrates the effects of the duration of the simulation on the accuracy of the results.

Example 4.1. The two models of Example 3.1 and Example 3.2 are considered. The mean waiting times of these models have been computed analytically and are exact. These models are now simulated using the simulation program and the obtained results are compared with the exact solution. For the warming-up time a setting of 10^4 time units is used and the simulation time is varied between 10^4 and $2 \cdot 10^6$ time units. For each setting the simulation is executed ten times and the results are averaged for a reliable measurement. The duration of the simulation is expressed in real-time and is measured as the time between starting and exiting the program. The command line version of the simulation program is used for obvious reasons. Figure 4.1 shows the results of these experiments.

From this figure a couple of observations can be made. Firstly, simulation of the model of Example 3.1 takes more time than simulation of the model of Example 3.2, whereas the simulation time parameters are the same in both cases. This can be observed from the horizontal axes in the graphs. The range of the axis of the left graph is more stretched than that of the right graph. The explanation for this can be contributed to the model properties. The switch-over times of the model of Example 3.1 are smaller than the switch-over times of the model of Example 3.2. This means that in the second model more time is elapsed when a server switch occurs compared to the first model, and thus that fewer events are needed to complete the simulation. Also, the arrival rates of the first model are lower while the service times of both models are approximately the same. Thus it occurs more often that the system is empty and that during this time period the server is merely switching.

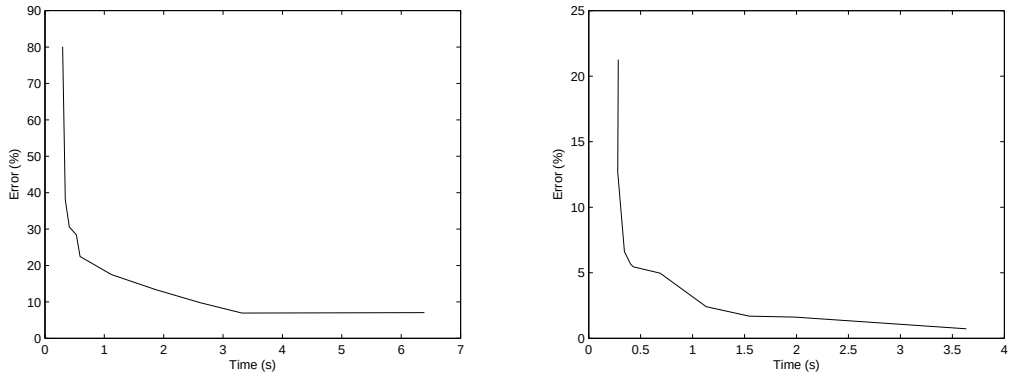


Figure 4.1. The relative error of the simulation results compared with the exact solution as a function of the simulation time, expressed in real-time. The graph on the left corresponds to the model of Example 3.1 and the graph on the right corresponds to the model of Example 3.2.

A second observation is that the error of the first model is higher than the error of the second model. This is not quite what one could expect when looking at the system parameters. Since the first model is symmetric and has service and switch-over processes with small variances, compared to the second model which has higher variances and is asymmetric, one could expect that the first system is 'easier' to simulate. With this is meant that the results of the simulation of the first model converge faster to exact solution than the second model. However, this is not the case. After simulation of $2 \cdot 10^6$ time units (6.4 seconds for the first model and 3.6 seconds for the second model) the relative error of the first model is 7.0% while the relative error of the second model is only 0.7%. This difference is the consequence of the offered load in combination with the service policy. While the load offered to the first system is 0.75 and the load offered to the second system is 0.9, the first system is closer to its stability limit. For the first system the stability condition reads $0.975 < 1$ and for the second system this condition is $0.9 < 1$. So the first system is evidently harder to simulate.

Finally, one can observe that the relative error reduces to zero eventually when the simulation time increases. This might not immediately be clear when looking at the left graph, but this is indeed the case when one simulates longer. For example, after simulating the first model for one minute an error of 1–2% is obtained and this error is reduced to 0.3–0.7% when simulating for ten minutes. When simulating the second model for the same amount of time, an error of 0.2–0.5% is obtained after one minute and an error of around 0.1% after ten minutes of simulation.

4.2.1 Effects of the service policy

In Example 4.1 it was shown that the relative error is a decreasing function of the simulation time. It was also shown that differences in model parameters contribute to differences in

the time it takes to simulate the model, and also to differences in the relative error. In this subsection the effects of varying the service policy on the relative error are analyzed, while keeping all other parameters the same. To this extend the switch-over times must be zero, because otherwise the offered load with respect to the system's stability limit is higher in systems with the k-limited service policy. As a consequence, the conservation law can now be used as a performance measure since it is easy to compute both analytically and by means of simulation. It has the additional advantage that it is the same for every service policy. The following example describes the test environment that is used to compare the different service policies.

Example 4.2. Consider a four-queue polling system where arrivals occur at all queues according to Poisson arrival processes of which the rates are the same for each queue. The service times at queue 1 are exponentially distributed with rate 2. The service times at queue 2 are gamma distributed with shape parameter 0.75 and scale parameter 2. The service times at queue 3 are uniformly distributed over the interval $[0, 2]$. The service times at queue 4 are deterministic distributed with value 2. The switch-over times are all zero. The simulation is done for various service policies and for different values of the total offered load (by changing the arrival rates) using a warming-up time of 10^5 time units and a simulation time of 10^7 time units. The simulations are repeated fifty times for each configuration and the results are each time compared with the exact conservation law. The averages are then taken to represent that configuration. The results are outlined in the graphs of Figure 4.2.

Aside from small fluctuations, which are inherent in simulation, the results are more or less consistent with each other. The relative error generally lies between 0.2–0.4% for values of ρ between 0.2 and 0.8. However the error is much higher when $\rho = 0.1$ or $\rho = 0.9$. This is in accordance with the theory that both in lightly and heavily loaded systems results become more inaccurate and thus that more time is needed to obtain the same level of accuracy compared with medium loaded systems. The conclusion that can be drawn from this example is that the service policy does not affect the accuracy of simulation in any way.

4.2.2 Effects of the number of queues

The required number of elementary operations of numerical techniques (e.g., the buffer occupancy method) is polynomial-bounded by the number of queues. In the following example the relation between the number of queues and the accuracy of simulation is investigated to find out whether a similar relation holds.

Example 4.3. Consider the model of Example 3.1 where the 1-limited service policy is used in a symmetric system setting. From the previous example it is clear that the service policy has no effect on the accuracy, and a symmetric design is chosen in order that all queues are equally important. The simulation is done for various number of queues, while keeping the load of the system fixed. This is achieved by changing the arrival rates. The simulation is performed twenty times for each configuration, using a warming-up time of

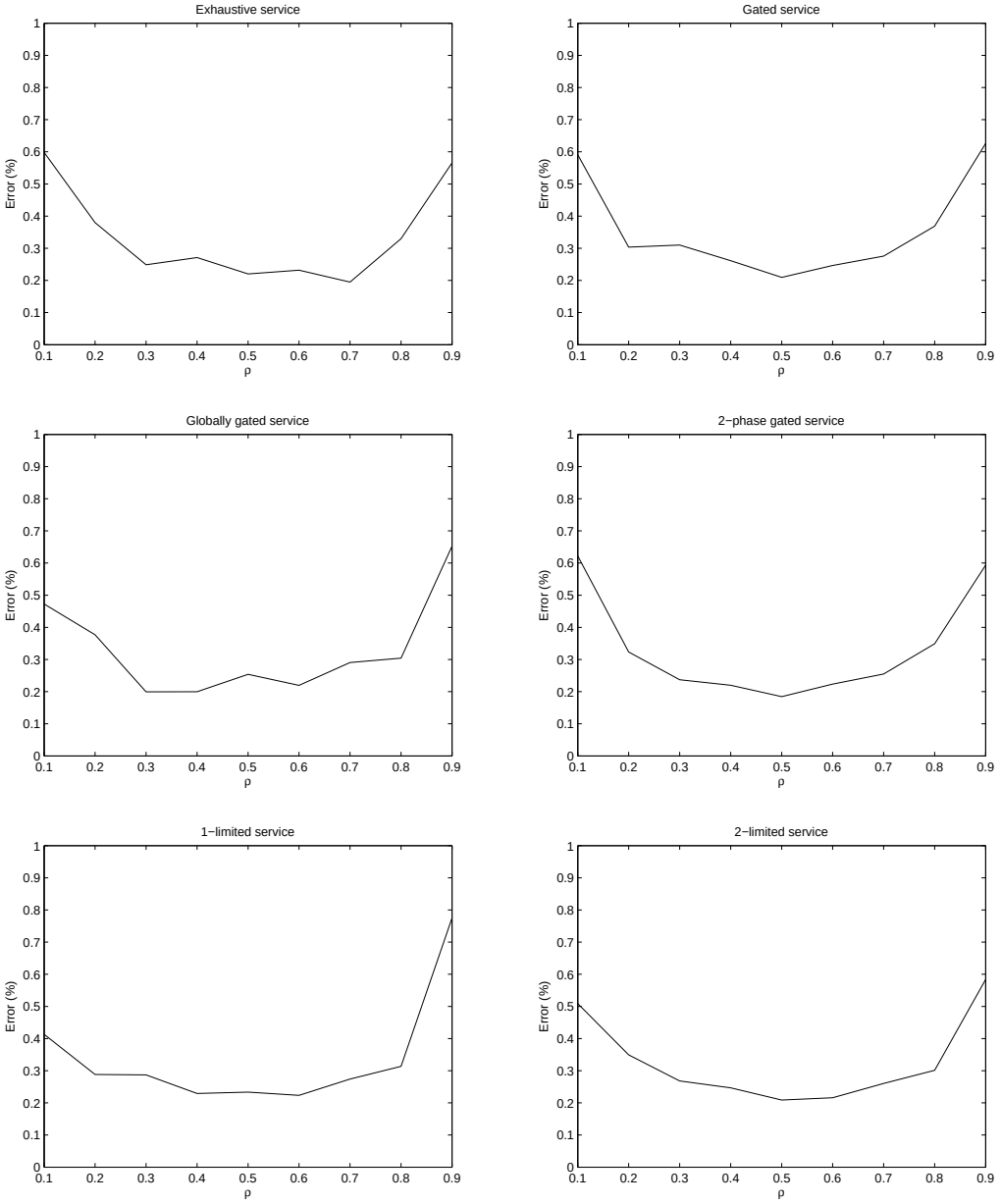


Figure 4.2. The relative error of the simulation results compared with the exact conservation law as a function of the load, for various service policies.

10^5 time units and a simulation time of 10^7 time units. The overall average waiting time obtained from the simulation is then compared with the exact solution. Figure 4.3 shows the results of these experiments.

From this figure it can be observed that the number of queues does not influence the accuracy of simulation. For example, the mean of the system with only one queue is ap-

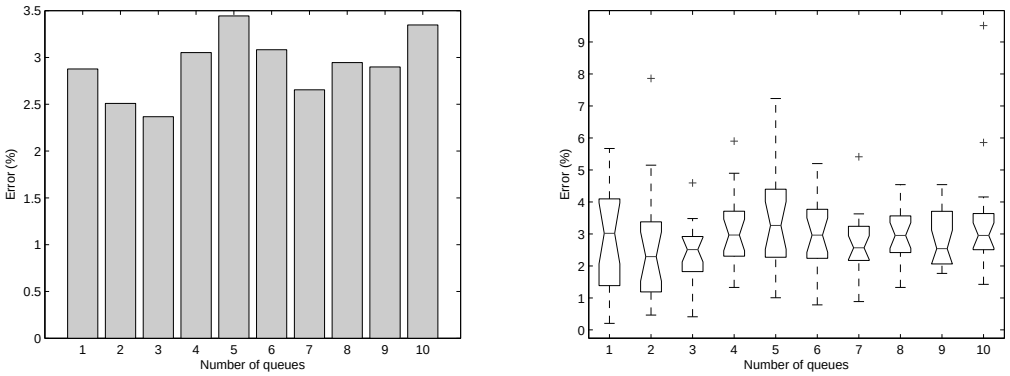


Figure 4.3. The relative error of the simulation results compared with the exact solution, as a function of the number of queues. The graph on the left shows the means of the results and the graph on the right shows boxplots of the same results.

proximately the same as the mean of the system with nine queues. The mean of the system with ten queues is a bit higher on the other hand, but this can be explained by the two outliers, as can be seen in the graph with the boxplots. Furthermore, all boxes overlap. So there is clearly no evidence to believe that the number of queues has an effect on the accuracy.

The insensitivity of the number of queues on the accuracy of simulation is a remarkable result, which has a striking consequence. There where at first it was advocated that simulation is inefficient and that results based on simulation are relatively inaccurate compared with numerical techniques, it now turns out that if the number of queues is large enough, simulation techniques are to be preferred to numerical techniques. As an illustration, if the model of Example 4.3 is simulated with hundred queues, while keeping the simulation time and all other parameters the same, a relative error is obtained that is only slightly above the errors presented in Figure 4.3. When the simulation time is doubled, then the results (both the mean and the boxplot) are on par with the model with nine queues. But, if the waiting times of a model with hundred queues are computed numerically with, e.g., the buffer occupancy method, it will take about 10^6 times as long as a model with only a few queues.

Remark 4.1. The simulation time needs to be doubled to obtain the same level of accuracy in case of the system with hundred queues. The reason for this is the following. The arrival rate at each queue is becoming smaller while the total switch-over time is becoming larger, as the number of queues increases. So during a cycle (i.e., the visit of the server along each queue) there are approximately the same number of arrivals, but the cycle length has increased. This means that fewer cycles fit in the total simulation time, and thus that each queue receives fewer arrivals. The accuracy of simulation is thus not 100% insensitive to the number of queues, but the influence is so small that it is negligible.

Bibliography

[1] O.J. Boxma and W.P. Groenendijk. Pseudo-conservation laws in cyclic-service systems. *Journal of Applied Probability*, 24(4):949–964, 1987.

[2] O.J. Boxma, H. Levy, and U. Yechiali. Cyclic reservation schemes for efficient operation of multiple-queue single-server systems. *Annals of Operations Research*, 35(3):187–208, 1992.

[3] Y.A. Bozer and M.M. Srinivasan. Tandem configurations for automated guided vehicle systems and the analysis of single vehicle loops. *IIE Transactions*, 23(1):72–82, 1991.

[4] M.S. van den Broek. Traffic signals: optimizing and analyzing traffic control systems. Master's thesis, Eindhoven University of Technology, 2004.

[5] W. Bux. Token-ring local-area networks and their performance. *Proceedings of the IEEE*, 77(2):238–256, 1989.

[6] W. Bux and H.L. Truong. Mean-delay approximation for cyclic service queueing systems. *Performance Evaluation*, 3(3):187–196, 1983.

[7] R.B. Cooper. Queues served in cyclic order: waiting times. *The Bell System Technical Journal*, 49(3):399–413, 1970.

[8] R.B. Cooper and G. Murray. Queues served in cyclic order. *The Bell System Technical Journal*, 48(3):675–689, 1969.

[9] C.F. Daganzo. Some properties of polling systems. *Queueing Systems*, 6(2):137–154, 1990.

[10] I.M. Dukhovnyy. An approximate model of motion of urban passenger transportation over annular routes. *Engineering Cybernetics*, 17(1):161–162, 1979.

[11] M. Eisenberg. Queues with periodic service and changeover time. *Operations Research*, 20(2):440–451, 1972.

[12] D.E. Everitt. A conservation-type law for the token ring with limited service. *British Telecom Technology Journal*, 4(2):51–61, 1986.

[13] M.J. Ferguson and Y. Aminetzah. Exact results for nonsymmetric token ring systems. *IEEE Transactions on Communications*, 33(3):223–321, 1985.

[14] C. Fricker and M.R. Jaïbi. Monotonicity and stability of periodic polling models. *Queueing Systems*, 15(1–4):211–238, 1994.

[15] B. Gamse and G.F. Newell. An analysis of elevator operation in moderate height buildings I: A single elevator. *Transportation Research Part B: Methodological*, 16(4):303–319, 1982.

[16] B. Gamse and G.F. Newell. An analysis of elevator operation in moderate height buildings II: Multiple elevators. *Transportation Research Part B: Methodological*, 16(4):321–335, 1982.

[17] A.G. Konheim, H. Levy, and M.M. Srinivasan. Descendant set: an efficient approach for the analysis of polling systems. *IEEE Transactions on Communications*, 42(2–4):1245–1253, 1994.

[18] A.G. Konheim and B. Meister. Waiting lines and times in a system with polling. *Journal of the Association for Computing Machinery*, 21(3):470–490, 1974.

[19] D.-S. Lee and B. Sengupta. An approximate analysis of a cyclic server queue with limited service and reservations. *Queueing Systems*, 11(1–2):153–178, 1992.

[20] K.K. Leung. Cyclic-service systems with probabilistically-limited service. *IEEE Journal on Selected Areas in Communications*, 9(2):185–193, 1991.

[21] K.K. Leung. Cyclic-service systems with non-preemptive, time-limited service. *IEEE Transactions on Communications*, 42(8):2521–2524, 1994.

[22] K.K. Leung and D.M. Lucantoni. Two vacation models for token-ring networks where service is controlled by timers. *Performance Evaluation*, 20(1–3):165–184, 1994.

[23] H. Levy. Optimization of polling systems: the fractional exhaustive service method. Technical report, Tel-Aviv University, 1988.

[24] H. Levy. Delay computation and dynamic behavior of non-symmetric polling systems. *Performance Evaluation*, 10(1):35–51, 1989.

[25] H. Levy and M. Sidi. Polling models: applications, modeling and optimization. *IEEE Transactions on Communications*, 38(10):1750–1760, 1991.

[26] Z. Liu and P. Nain. Optimal scheduling in some multi-queue single-server systems. *IEEE Transactions on Automatic Control*, 37(2):247–252, 1992.

[27] C. Mack. The efficiency of N machines uni-directionally patrolled by one operative when walking time is constant and repair times are variable. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 19(1):173–178, 1957.

[28] C. Mack, T. Murphy, and N.L. Webb. The efficiency of N machines uni-directionally patrolled by one operative when walking time and repair times are constants. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 19(1):166–172, 1957.

[29] R.D. van der Mei. Distribution of the delay in polling systems in heavy traffic. *Performance Evaluation*, 38(2):133–148, 1999.

[30] R.D. van der Mei. Polling systems with switch-over times under heavy load: Moments of the delay. *Queueing Systems*, 36(4):381–404, 2000.

[31] R.D. van der Mei. Towards a unifying theory on branching-type polling models in heavy traffic. To appear in Queueing Systems, 2007.

[32] R.D. van der Mei and H. Levy. Expected delay analysis of polling systems in heavy traffic. *Advances in Applied Probability*, 30(2):586–602, 1998.

[33] R.D. van der Mei and A. Roubos. Dynamic bandwidth allocation in ethernet passive optical networks. In preparation, 2007.

[34] S. Nahmias and M.H. Rothkopf. Stochastic models of internal mail delivery systems. *Management Science*, 30(9):1113–1120, 1984.

[35] G.F. Newell. Properties of vehicle-actuated signals I: One-way streets. *Transportation Science*, 3(1):30–52, 1969.

[36] G.F. Newell and E.E. Osuna. Properties of vehicle-actuated signals II: Two-way streets. *Transportation Science*, 3(2):99–125, 1969.

[37] J.A.C. Resing. Polling systems and multitype branching processes. *Queueing Systems*, 13(4):409–426, 1993.

[38] D. Sarkar and W.I. Zangwill. Expected waiting time for nonsymmetric cyclic queueing systems—exact results and applications. *Management Science*, 35(12):1463–1474, 1989.

[39] E. de Souza e Silva, H.R. Gail, and R.R. Muntz. Polling systems with server timeouts and their application to token passing networks. *IEEE/ACM Transactions on Networking*, 3(5):560–575, 1995.

[40] M.M. Srinivasan, H. Levy, and A.G. Konheim. The individual station technique for the analysis of polling systems. *Naval Research Logistics*, 43(1):79–101, 1996.

[41] H. Takagi. *Analysis of Polling Systems*. The MIT Press, Cambridge, Massachusetts, 1986.

[42] H. Takagi. Queuing analysis of polling models. *ACM Computing Surveys*, 20(1):5–28, 1988.

[43] H. Takagi. Queueing analysis of polling models: an update. In *Stochastic Analysis of Computer and Communication Systems*, pages 267–318, Amsterdam, North-Holland, 1990. Elsevier Science Publishers B. V.

[44] H. Takagi. Application of polling models to computer networks. *Computer Networks and ISDN Systems*, 22(3):193–211, 1991.

[45] H. Takagi. Queueing analysis of polling models: progress in 1990-1994. In *Frontiers in Queueing: Models and Applications in Science and Technology*, pages 119–146, Boca Raton, Florida, 1997. CRC Press.

[46] H. Takagi. Analysis and application of polling models. In *Performance Evaluation: Origins and Directions*, pages 423–442, Berlin, Heidelberg, 2000. Springer-Verlag.

[47] V.M. Vishnevskii and O.V. Semenova. Mathematical methods to study the polling systems. *Automation and Remote Control*, 67(2):173–220, 2006.

[48] E.M.M. Winands. *Polling, Production & Priorities*. PhD thesis, Eindhoven University of Technology, 2007.

[49] A.O. Zaghloul and H.G. Perros. Approximate analysis of a discrete-time polling system with bursty arrivals. In *Proceedings of the IFIP TC6 Task Group/WG6.4 International Workshop on Performance of Communication Systems*, pages 61–80, Amsterdam, North-Holland, 1993. North-Holland Publishing Co.