

Chapter 1

MULTIACCESS IN CABLE NETWORKS

This book proposes and analyses delay models for communication over a shared channel by means of a reservation procedure. In this introductory chapter, we discuss in general terms the main concepts for multiaccess communication: Contention resolution and reservation procedures. We then turn to cable networks and describe in some detail the current protocols to transmit data. Moreover, we give examples, obtained by complex system simulations, of the specific performance issues that arise in the evaluation of these networks. These examples have strongly motivated our research effort, in which we aim at complementing the simulated results with analytical models and calculations.

1.1 Multiaccess Communication

Multiaccess communication is a well established research topic, see, e.g. Bertsekas and Gallager [14], Hayes [77], or Tanenbaum [160]. The central setting of multiaccess communication is that a number of entities use the same communication channel to transmit messages. This setting will be familiar to most of us from everyday experience. We all use the same channel (the air) for talking. One of the issues associated with multiaccess communication is then also clear from daily life: Speech becomes unintelligible if several people are talking at the same instant.

The examples below serve to illustrate the wide variety of application areas in machine-to-machine communication in which multiaccess communication is relevant.

EXAMPLE 1.1 Computer networks. In recent years, computers have become interconnected to form computer networks. This enables them to exchange messages. Frequently, the link between two computers overlaps with similar links between other computers. This is the case in wired and wireless

local area networks with either the Ethernet [122] or the IEEE 802.11 standard [106], or in access networks with, e.g. the DOCSIS [58], the IEEE 802.14 [80], or the DVB-DAVIC [59] standard.

Whatever the precise nature of the computer network and its links, transmission of different messages over the same link at the same time usually causes interference which in turn causes message loss. Hence, mechanisms must be implemented to achieve satisfactory link sharing.

A popular way to achieve this, and indeed the method implemented in the protocols mentioned above, is time division: A message is only transmitted successfully if it does not coincide in time with another message transmitted over the same link.

EXAMPLE 1.2 Radio frequency tagging. It is likely that in the near future many consumer goods will be labeled with identification tags that can be read automatically from a distance without line-of-sight, see Finkenzeller [67] or Law et al. [107]. Tagging is useful for object tracking, as in luggage handling in airports, or for automatic purchase handling in shops.

Tags are low-cost, passive devices. However, they can take energy from certain radio frequency signals emitted by a tag-reader. The power obtained from this signal allows them to respond, and the response will typically contain an identification of the tagged object. Upon receipt of this response, the tag-reader can retrieve information relevant to the object from a database. This information can then be used for a routing decision, as in luggage handling systems. Alternatively, this information can be a price, so that the purchase can be handled automatically.

If multiple tagged objects respond simultaneously to the activating radio signal, the responses will interfere and the object identifications will be unintelligible to the tag-reader.

EXAMPLE 1.3 On chip communication. Systems on Chips (SoCs) are integrated circuits that offer the functionality of a complete system on a single chip; examples of SoCs are single-chip televisions, MPEG encoders, and so on. These SoCs comprise an ever increasing number of basic devices, see, e.g. Jantsch and Tenhunen [?]. These devices form the building blocks that are combined to implement the full functionality of the system.

The devices must be interconnected. Currently, most SoCs have a shared medium architecture for their interconnection network, most often a backplane bus, see, e.g. Benini and de Micheli [13]. In this architecture, there is one communication channel, the bus, that is shared by all the devices. In order to transmit a message, a device must first gain bus mastership, because the bus does not support simultaneously transmitted messages. Hence, bus arbitration mechanisms are necessary when several devices attempt to use the bus simultaneously.

In these three examples, there is contention for the use of the communication channel between the various nodes in the network. Conflicts cause message loss, so that there is a need for some form of conflict resolution in order to guarantee that every node eventually gets hold of the channel and can successfully transmit its message. Ideally, the conflict resolution is organised in such a way that successful message transmission is achieved in the shortest possible time.

How to organise the conflict resolution depends on the context. Again, examples from daily life may illustrate the point. For example, in dinner conversation, it is all right to just start talking when we think of something to say. But in a business meeting it might be the chairman's job to determine who speaks next.

Basic methods to organise access in machine communication resemble these simple examples. Thus in some protocols it is allowed that stations just attempt transmission without consideration of other stations. Stations then continue to transmit their message until it is successfully transmitted. Other protocols dictate a more systematic approach in which stations take turns. Between these extremes there is a whole spectrum of possible protocols. We now give a more systematic account of these access methods.

1.2 Methods for Multiaccess

The standard references on computer networks [14, 77, 160] cover the basic methods for multiaccess: Frequency Division Multiple Access (FDMA), Code Division Multiple Access (CDMA), and Time Division Multiple Access (TDMA). With FDMA, stations are assigned different frequency channels, so that conflicts are avoided as collisions in a frequency band can no longer occur. With CDMA, signals from stations are encoded in such a way that conflicts with messages from other stations are experienced as 'white noise'. Thus, conflicts can be resolved as the noise can be removed using the appropriate decoders. We will be concerned exclusively with TDMA networks, in which simultaneously transmitted messages are always corrupted. Therefore, the channel must be divided over time among the different stations.

There are a number of techniques for implementing TDMA. Hayes [77], Sect. 2.7.4, considers polling, token passing, and random access protocols to achieve TDMA. In polling, a central server addresses all stations in the network in turn. A station that receives a polling message can interrupt this polling cycle to transmit messages. After this transmission, the polling cycle continues. In token passing, there is no central entity that organises the access. Rather, the stations pass a token among each other. This token is a licence to transmit, so that the station in possession of the token can transmit its messages. Upon completion of the transmission, it passes the token on to the next station.

Polling and token passing implement a systematic approach to multiaccess, in which stations are polled, in fixed order, for data transmission. Conflicts are thus avoided. The performance of these systematic approaches depends on the characteristics of the network and the communication needs of the stations. Generally, these techniques are less suitable if there are many stations in the network and if their communication patterns are very bursty. In these circumstances, one prefers random access protocols, see Tsybakov [163].

The best known random access protocol is ALOHA, see Roberts [149]. Using ALOHA, stations transmit their messages immediately, without any coordination with other stations. Therefore, conflicts can arise, and it is necessary that the stations obtain some form of feedback concerning the success of their transmission. This feedback can be obtained through an acknowledgement sent by a receiving station or by a central scheduler. Alternatively, the station can listen to the channel and detect transmission conflicts.

If a transmission error occurs, the station waits a random ‘back-off’ time and then retransmits its message. This procedure is repeated, possibly with increasing back-off times, until successful transmission. There are many variants of ALOHA that differ as to how this random time is exactly determined. The best of these, slotted stabilised ALOHA, achieves a channel utilisation of $e^{-1} \approx 36\%$ for an open, infinite-population model with Poisson arrivals, see Bertsekas and Gallager [14], Sect. 4.2.3.

In case the stations have the ability to quickly check channel status and interrupt the transmission of their messages, improvements of ALOHA are possible. These improvements are called Carrier Sense Multiple Access (CSMA) or Carrier Sense Multiple Access with Collision Detection (CSMA-CD). Throughputs for these improvements depend on the speed with which transmission errors can be detected, but are much larger than the 36% that can be achieved with slotted stabilised ALOHA. In fact, if transmission errors can be detected infinitely fast, throughputs arbitrarily close to 100% can be achieved via CSMA-CD, which explains the wide-spread popularity of this protocol.

If it is impossible to quickly detect transmission errors, more sophisticated random access protocols are preferable to ALOHA. These algorithms are based on so-called contention trees as introduced independently by Tsybakov and Mikhailov [164] and Capetanakis [30]. Using these techniques, the channel utilisation can be increased to approximately 49%, see Tsybakov and Mikhailov [165] and Mosely and Humblet [127]. We defer a more elaborate introduction of contention trees until Sect. 2.1.

Random access protocols can be used to directly transmit a message. However, they can also be used somewhat differently, as a signal to indicate the intention to transmit a message, i.e. as a request message for data transmission. Once this intention has been successfully transmitted, the message itself can be transmitted without the risk of being corrupted by interrupting mes-

sages. These approaches are known as reservation or request-grant procedures. In such procedures, stations request access to the shared medium by sending request messages in contention with other stations. The request messages will be small, relative to the actual message to be transmitted, and run the risk of being lost due to collisions with other requests. Hence, some form of conflict resolution is needed to ensure that a request will eventually get through. However, once the request gets through, the station will, in due time, be granted exclusive access to the channel so that it can transmit its messages without colliding.

Reservation procedures have a clear advantage over transmitting complete messages in contention with other messages: Only the small request messages run the risk of being lost. Therefore, these reservation procedures have the promise of efficient and fast data transmission. Not surprisingly, reservation procedures have been extensively investigated and have found application in many computer communication protocols.

In this book we develop analytical models for studying the delay in reservation procedures. Many aspects of these models are relevant to all protocols that implement conflict resolution. We are, however, motivated in particular by the use of reservation procedures in cable networks that involve contention trees. The models in this book are all developed to deal with specific issues that arise in this context. We now first discuss cable networks and their reservation procedures, and then turn to these modelling issues.

1.3 Data Transfer in Cable Networks

A schematic view of a cable network is given in Fig. 1.1. In order to transmit messages the stations must use the upstream channel, which is shared with the other stations. Messages can only be received via the downstream channel. As stations cannot sense the upstream channel, they cannot communicate with each other directly nor can they check the success of their own transmissions. To provide the essential feedback, cable networks incorporate a central scheduler referred to as the Head-End. The Head-End continuously senses the

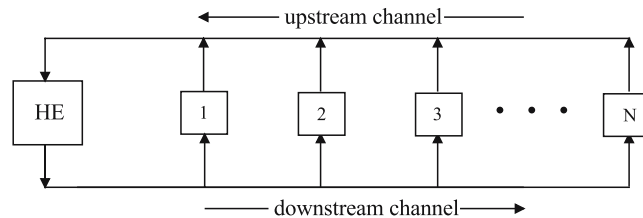


Fig. 1.1. Schematic view of a cable access network with N stations connected to the Head-End (HE)

upstream channel, observes the success or failure of transmitted messages, and informs the stations about this by broadcasting status reports via the downstream channel.

A system's account of the recent upgrade of cable networks to enable interactive services is given in, e.g. [15, 56, 57]. The multiaccess related details can be found in, e.g. [40, 73, 74, 145, 150]. A number of standard protocols for data transfer in cable networks have emerged and we review the issues that are most relevant to this book. Important protocols are IEEE 802.14 [80], DVB-DAVIC [59], and DOCSIS [58]; DOCSIS is by now the generally accepted standard. These standards describe in great detail the physical elements of the network, the channel medium, the formats of the signals, the conversion of packets into signals and vice versa, the error correction, etc. We largely ignore the physical details of data transmission and concentrate on the medium access control (MAC) layer.

Computer communication is commonly organised as a stack of layers, the OSI stack, see, e.g. [14], Sect. 1.3.2. Each layer in this stack offers specific functionalities to the above layers that are needed to establish a connection. In this system view, the data link layer is situated between the physical layer and the network layer. The MAC sublayer is considered as lower part of the data link control layer and regulates the multiaccess channel.

The protocols describe a variety of methods for data transmission, and differ as to which methods are allowed. Generally, it is possible to transmit data both directly and via the reservation procedure. We will concentrate on the analysis of the reservation procedure and consider the following situation described in, e.g. [59, 80]. Stations request access to the upstream channel for data transmission by transmitting a request message. These requests are in contention with the requests from other stations and are transmitted in time slots that have been specifically designated by the Head-End as request slots. As requests can collide with other requests, some form of conflict resolution must be implemented in order to guarantee that every request will eventually get through to the Head-End. We will focus on the situation in which conflicts between request messages are resolved by means of contention trees. These are introduced in Sect. 2.1 and are extensively analysed in Part II of this monograph. After a successful request, data transfer follows in reserved 'data-transmission' slots, not in contention with other stations. These transmission slots are assigned by the Head-End by means of a message broadcast via the downstream channel.

Stations communicate via an exchange of quanta of information called packets. Such a packet can be a request packet or a data packet. Usually, a request packet is smaller than a data packet, and we consider the specific situation in which it is three times as small as a data packet. The upstream channel is time slotted. Each time slot can contain exactly one data packet. Alternatively, three request packets can be fitted into one time slot.

This notion of a data packet as the basic unity of communication allows for a more precise description of the reservation procedure described above. The rule in this reservation procedure is that a station can make a request for data transmission as soon as it has data to transmit. The request message contains a field, called the request size, to indicate the number of data packets for which transmission is requested. The request size equals the number of data packets actually ready for transmission at the instant of the request attempt. Request messages may be lost due to a conflict with other requests. In this case another request message must be sent. This new request message is identical to its predecessor, except for the value of the request size field: The request size may be increased if the number of packets ready for transmission has increased since the previous request attempt. This property is called ‘partially gated’ in [14], Sect. 3.5.2.

Whether a time slot will be used for data or for requests is decided by the Head-End, which periodically broadcasts the use of the next sequence of time slots to all stations in the network. If a time slot is to be used for data, the Head-End will also assign it to one of the stations, which can then use it, contention free, to transmit data. If a time slot is to be used for requests, the Head-End can restrict the use of this request slot to subsets of stations satisfying various criteria. The decision whether to use a slot as either a request or a data-transmission slot is usually taken simultaneously for groups of 18 consecutive slots, called frames. The duration of such a frame is approximately 3 ms. The Head-End employs a scheduling strategy to determine the frame layout and in particular the number of request slots and the number of data-transmission slots. After the layout has been determined, it is broadcast to all the stations in the network.

It is another important characteristic of cable networks that these scheduling decisions concerning the frame layout must be taken well before the actual start of the frame to which they correspond. There are a number of reasons for this. Firstly, cable networks are large and the number of stations connected to one Head-End is typically in the order of 100–1,000. As a consequence, such networks can cover a wide area, and the distance between stations and Head-End can be quite substantial. Hence, there is a non-negligible propagation delay involved in transmitting a message from a Head-End to a station and vice versa. Secondly, messages in cable networks are interleaved. This means that they are spread out over time, so as to make the error correction more effective against burst errors. However, this also increases the length of the message: The time between the start of a message and its end. Thirdly, it requires processing time at the Head-End to calculate the schedules and at the station to process and interpret the schedules.

The length of the scheduling message (including interleaving) plus the propagation times plus the processing times at both Head-End and station is non-negligible and is at least about equal to the frame duration. Depending on the

implementation details, it can be larger, and information from the current frame can only be used some frames later. Therefore, scheduling decisions must be taken well in advance. In particular, there is a delay of at least some frames before the data transfer corresponding to a request can actually be carried out.

In the remainder of this book, this delay will often be referred to as the round-trip time. It is especially relevant for the model considered in Chap. 8.

1.4 Performance Analysis of Cable Networks

A common, simplifying, attempt to understand the performance achieved in multiaccess systems is to view the shared channel as a kind of central ‘server’ that has to serve a stream of incoming messages, see, e.g. [14], Sect. 4.1. Such models are known as open models, as the packets in the arrival stream are not associated with specific originating stations. Using this approach, the analysis of the delay in cable networks can then proceed along classical lines.

However, the discussion in Sect. 1.3 has emphasised a number of characteristics that show that such a view has its limitations for the performance analysis of cable networks. These limitations emerge mainly as a consequence of the reservation procedure. Firstly, there are at most a finite number of stations that can enter the contention procedure to transmit a request message. Thus, one can expect that there is a finite-population effect that will show up in medium to high load conditions, which is not captured by open models. Secondly, the server must be shared by the reservation process and the data-transmission process and the optimal tuning of these processes is complicated by the propagation delay. The determination of the bandwidth allocated to either process requires a scheduling policy, which does not have a counterpart in the simplifying model described above.

As there are hardly any accurate models to assess the performance of cable networks, their evaluation is usually carried out by means of simulations. We have drawn in particular upon the simulation platform described in Kwaaitaal [104] and Pronk and De Jong [144]. This platform was used to investigate the performance of cable networks in [78, 145]. In the simulations, stations generate data packets according to a Poisson process. These are transmitted over the cable network via a reservation procedure using contention trees. Performance figures are recorded concerning queue sizes and packet delay, yielding important insights. Moreover, it is possible to compare various slot scheduling algorithms that are used in the Head-End.

We now review some of these simulation results. They demonstrate that the finite-population effect and the scheduling effect do indeed manifest themselves as major issues in the performance analysis of cable networks. This sets the main objective for this book: To formulate and analyse performance models that incorporate the finite-population effect and scheduling policies. These models will then enable us to complement the simulations with analytical calculations.

1.4.1 A Finite-Population Effect

In a first experiment, we compare the mean packet delay in three different scenarios. In each scenario the stations generate data, identically and independently, according to a Poisson process, but the scenarios differ in that the number of stations in the network is varied. If we would use standard open queueing models to predict the average delay, then there would be little difference between the three scenarios: As the superposition of a number of Poisson processes is again a Poisson process, only the total traffic volume matters here.

Performance curves for this experiment are presented in Fig. 1.2 for the cases that there are 50, 100, and 200 stations in the network. These curves display the average delay of a transmitted packet against the total traffic intensity in each of the three scenarios. The performance curves for the three scenarios differ considerably, and show that standard open models are not directly applicable in this situation. The simulation results thus point to a finite-population effect: The number of stations in the network strongly affects packet delay.

This finite-population effect is observed in many system simulations of cable networks. In Pronk et al. [145], Sect. 4.2, it is shown that this finite-population effect also shows up in cable networks with other data-transfer modes not considered here. Moreover, Golmie et al. [74], Fig. 17, report a similar effect.

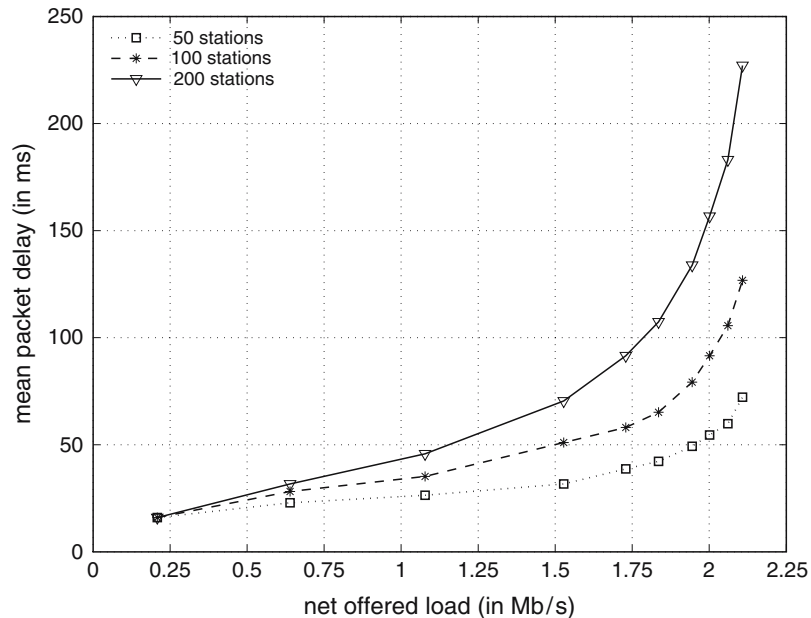


Fig. 1.2. Delay vs. load curves for three different network scenarios, in which networks have different numbers of stations

There is another salient feature in Fig. 1.2, extensively discussed in Denteneer and Pronk [55], which concerns the capacity of the network. As a preliminary fact, we mention that the capacity of the contention tree is approximately $\log(3)$, see Sect. 3.3.1. This means that, on average, $\log(3)$ requests can be processed during one time slot consisting of three request slots. The maximum total traffic intensity in the simulations is approximately equal to 2 Mbps, which corresponds to a traffic intensity of 16 data packets per frame, for frames of 18 slots as described in Sect. 1.3.

If the open queueing model is adopted as a model for data transfer in a cable network, then it is easy to see that the transfer of each packet requires on average $(1 + 1/\log(3))$ time slots: $1/\log(3)$ for the request and 1 for the packet transfer itself, cf. [14], (4.59). This would limit the capacity of the network to $18/(1 + 1/\log(3)) \approx 9.5$ data packets per frame. However, this upper bound on the capacity is well below the maximum value of 16 packets per frame, mentioned above. This phenomenon can be explained if we consider a closed queueing system. In this case packets originate from a particular station, and it is well possible that stations ask for the transmission of more than one data packet in one request. This further underlines that the models should reflect that there are only a finite number of stations active in the network.

1.4.2 A Scheduling Effect

A second experiment is concerned with the scheduling of time slots as either request slots or data-transmission slots. Here, we fixed a network scenario and considered three different schedules to designate the use of the time slots in a frame. In each of these schedules, scheduling decisions are frame-based. The schedules originate from Pronk et al. [145] and each schedule guarantees a different minimum number of time slots per frame for the request procedure. These slots are devoted to handling requests in contention. The other time slots in a frame are used for data transmission. If there are not enough data packets to use the whole frame, the remaining slots are again used for handling requests in contention.

In Fig. 1.3, we have displayed the simulated performance curves for three distinct schedules. In the simulations, the number of slots guaranteed for the request process equalled 3, 6 and 9, respectively, but the actual values are not important for the discussion. Figure 1.3 shows that the different schedules result in very similar performance at low loads. However, at high loads, the performance curves deviate considerably. Hence, it is apparent that there is a scheduling effect. Moreover, no schedule dominates the others. Rather, each schedule is best in a selected subinterval of the traffic load.

This scheduling effect is not specific to the details of our simulations and has also been observed in other simulations of cable networks. For instance, in Sala et al. [150] system simulations are carried out using ALOHA for requests.

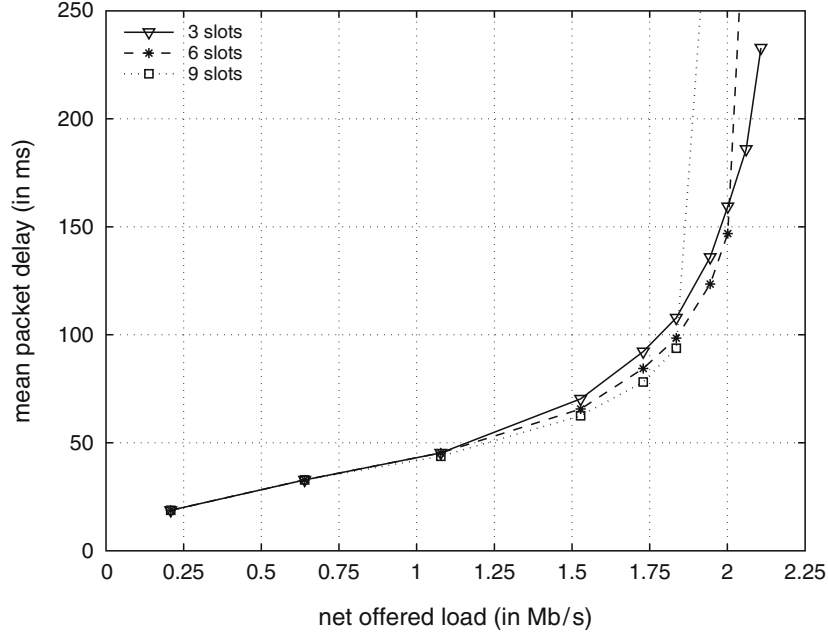


Fig. 1.3. Delay vs. load curves for three different ways to organise the contention process

First, they tested a greedy scheduling strategy, in which slots are only given to the request process if there are no more data to transmit, i.e. if the data queue is empty. It was observed that this strategy resulted in cyclic behaviour of the data queue. Next, it was established experimentally that performance improves if a certain minimum number of slots is guaranteed to requests. Also Golmie et al. [73] consider various schedules for bandwidth allocation to either requests or data transmission. Their results are similar and further stress the relevance of these schedules.

1.5 Outline

Both the finite-population effect and the scheduling effect are prominent features of delay in cable networks as shown by the simulation results reviewed in Sect. 1.4. However, neither has been captured in the performance models for cable networks that have been proposed to date. In this book, we attempt to fill the gap and describe models that account for these effects.

Another way to view this book is suggested by the title and generalises the scope from cable networks to reservation procedures and queueing models. Thus, this book is about the delay analysis of the reservation procedure with contention trees. Our interest is focused on the finite population case and a communication channel with substantial round-trip times. We are now in a position to give an outline of this book.

Part I: Prologue

In Part I, we review the basic notions of multiaccess, cable networks, as well as the various key models that we build upon in the remainder of the book. In fact, there are four fundamental points of departure for the models in this book: Contention trees, the repairman model, the bulk service queue, and tandem queues. As such, the relevance of this book is not limited to cable networks as all models have found (or might find) application in other domains. The key models are reviewed in Chap. 2.

Part II: Contention Trees

The first basic notion is the Capetanakis-Tsybakov contention tree introduced in Capetanakis [30] and Tsybakov and Mikhailov [164]. Algorithms based on contention trees are among the best known algorithms for conflict resolution; we defer a more detailed description of contention trees until Sect. 2.1, where we also give a brief introduction to the contention tree literature.

Part II presents our own contributions to the analysis of contention trees. In Chap. 3, we give a mathematical account of the contention trees and present two formal ways to describe their evolutions. Both ways are based on the complete m -ary tree, also considered in Capetanakis [30] and Kaplan and Gulko [93]. These models are used to analyse a number of basic properties of contention trees. In Sect. 3.3.1 we consider statistics associated with the length of the contention tree, and in Sect. 3.3.3 we review a modification of the standard algorithm, named skipped level trees, which improves the standard contention tree algorithm with respect to the time needed to resolve all conflicts.

We consider the success instants of a contention tree in Sect. 3.3.2. The success instant measures the time from the start of the contention tree until a successful conflict resolution and is thus related to the delay when using contention trees. We use the models to give an expression for the marginal distribution of the success instant, see Theorem 3.3, and to prove that this expression is asymptotically exact when the number of contenders tends to infinity, see Theorem 3.4.

In Chap. 4, we extend the discussion to sequences of contention trees as they are used in practice in dynamic environments. In these environments, stations become inactive after having successfully transmitted their requests. However, as the contention tree is being processed, inactive stations can become active again when they have data to transmit. Hence, the basic contention tree algorithm must be complemented with a ‘channel access protocol’ or ‘first transmission rule’: A set of rules to regulate the behaviour of newcomers. In Sect. 4.2, we review the standard channel access protocols: Free access and blocked access. Section 4.2.1 describes an improvement of these standard protocols referred to as scheduled access. The scheduled access protocol improves the basic access protocols both in terms of capacity and delay properties.

We then turn to the request delay, defined as the delay experienced by requests made in contention via a contention tree. The complicated distributions of quantities that depend on the contention tree algorithm preclude a complete analytical treatment. However, the properties of the contention trees reviewed in Chap. 3 suggest that queueing approximations are appropriate for contention trees when used in dynamic environments. Moreover, the finite-population effect suggests that closed queueing models are appropriate.

The remainder of Chap. 4 is devoted to this theme and investigates closed queueing systems to approximate the request delay for the three contention tree protocols. It discusses a number of modifications of the so-called repairman model and gives an analysis of the associated sojourn times, either exactly or heuristically. The proposed modifications pertain to the service discipline. Next to the First Come First Served (FCFS) service discipline, as in the standard version of the repairman model, we also consider Random Order of Service (ROS), Gated Random Order of Service (GROS), and Gated Partial Random Order of Service (GPROS).

These modifications are all motivated by the various channel access protocols that complement the basic contention tree algorithm. We will argue that ROS is appropriate for contention trees with free access, that GROS is appropriate for contention trees with blocked access, and that GPROS is appropriate for contention trees with scheduled access. These observations are supported by simulations, in which we show that the analytical expressions for the first moments of the sojourn time in these repairman models give excellent approximations to the corresponding moments of the request delay for contention trees obtained by simulation.

The repairman model with GROS does not belong to the class of product-form networks, see [10], and we only give a heuristic analysis of its delay properties in Chap. 4. Motivated by the fact that cable networks are usually large, we set out for a more precise mathematical analysis in Chap. 5 using asymptotic analysis. This analysis is utilised to establish the limiting distribution of the request delay, see Theorem 5.2.

Part III: The Bulk Service Queue

In Part III of this book, we turn to the data-transmission delay: The delay due to the queueing of packets at the central scheduler, once a request has been successfully transmitted. We model this part of the reservation procedure by means of the bulk service queue. This is a discrete-time queueing model in which a central server periodically serves a fixed number of customers that arrive according to some random process.

A first introduction to the bulk service queue is given in Sect. 2.3. In Chap. 6 we give an extensive historical introduction of this queueing model, including the main techniques to solve for the stationary distribution. In our treatment, we give the rigorous mathematics, and provide guidelines for the numerical work.

The bulk service queue is introduced in particular to provide a mathematical foundation to analyse the scheduling effect, described in Sect. 1.4.2. The classical bulk service queue can be used to model a simple schedule, in which a fixed fraction of each frame is devoted to the request process. This is a wasteful schedule, as it potentially leaves capacity unused. In Chap. 7, we formulate a flexible-boundary model to represent a better scheduling policy. This schedule was also used in Sect. 1.4.2, and guarantees a fixed minimum amount of slots per frame for the request process. We show that this model can be solved by means of the classical techniques from Chap. 6. However, it fails to accurately capture the scheduling effect, as it does not account for the round-trip times on the communication channel.

Motivated by this shortcoming, we propose to modify the classical bulk service queue by including the delay as another additional element. Due to this delay, arriving packets can only be ‘served’ after some delay period. We give a thorough motivation for the delayed model in Sect. 2.3 and its detailed analysis is reported in Chap. 8. Although the delayed bulk service queue defines a higher-dimensional Markov chain, we are still able to solve for the stationary distribution of the queue, but the solution hinges on complicated numerical procedures. However, we are able to obtain better insight in the expected queue size by exploiting a method of Kingman [95] that leads to an insightful expression for the expected data-queue size as a function of moments of the arrival distribution, a term related to the idle time, and an auto-correlation term. We then bound the term related to the idle time, and apply a heuristic argument to approximate the correlation term, so that we are able to approximate the expected data-queue size of the delayed bulk service queue.

The approximating bounds are complemented with simulations to establish some interesting properties of the expected data-queue size. Firstly, the expected data-queue size is increasing with the delay. Secondly, and rather remarkably, the expected data-queue size is not monotonic in the traffic intensity, so that a larger traffic volume may actually result in a substantially lower expected data-queue size. Most importantly, however, the delayed bulk service queue aids in the formulation of good scheduling policies to split the bandwidth between the request and data-transmission processes. One such scheduling policy is obtained by a minimisation of the expected queue size with respect to the amount of capacity guaranteed to the request process. A second, more adaptive, schedule is derived by a minimisation of the burstiness of the arrival process, which can be controlled through the request process. The latter property makes the delayed bulk service queue suitable as a model for delay in cable networks, see Sect. 1.4.2.

Part IV: Tandem Queues with Shared Service Capacity

The approaches from Part II and III enable a decompositional approach to analyse the performance of reservation procedures, with separate models for the reservation phase and the data-transmission phase. For an integrated approach, we introduce tandem queues. In the tandem queue, users arrive at a first server, possibly queueing up if the server is busy. Upon service completion at this first server, they move on to a second server, again queueing in front of this server if this second server is busy. After being served at the second server, the users leave the system. In the classical tandem queue, the servers work independently of each other with a fixed service rate. We deviate from this classical model in that the servers in our model share a common service capacity, and this necessitates a service discipline that allocates the service capacity to the servers.

In Chap. 9 we give a detailed treatment of the two-stage tandem queue with shared service capacity. In order to do so, we need to make some rather restrictive assumptions. We assume that users arrive to the request queue according to a Poisson process, and that they require exponentially distributed service times at both queues. Under these assumptions, the tandem queue with shared service capacity gives rise to a two-dimensional Markov process that can be analysed using the theory of *boundary value problems*. A pioneering study of this type of Markov processes is the one of Malyshev [115], whose technique was introduced to queueing theory by Fayolle and Iasnogorodski [61]. We show that the problem of finding the generating function of the joint stationary queue length distribution can be reduced to a Riemann–Hilbert boundary value problem.

Starting from the solution of the boundary value problem, we consider the issues that arise when calculating performance measures like the mean queue size and the fraction of time a queue is empty.

In Chap. 10 we present a more general model of a *two-station network with shared service capacity*. After receiving service at a server, a user either joins the queue of the same server, joins the queue of the other server, or leaves the system, each with a given probability. Users require exponential service times at each server. This general network model covers both the model of Fayolle and Iasnogorodski [61] and the two-stage tandem queue with shared service capacity as special cases. We show that the general model can also be solved using the theory of boundary value problems.

Part V: Epilogue

In Chap. 11, we return to our point of departure and reconsider the reservation procedure in a cable network that motivated our work. In particular, we focus on the delay in cable networks and use the models and their analyses to approximate the total average packet delay. We compare this approximation with

the delay figures that are obtained in system simulations of cable networks. Moreover, we use the approximation to quantify the effects of suggestions for scheduling made in this book. We conclude the book with a brief review of open questions for further research.

1.6 Selected Bibliography

As acknowledged in our foreword, this book is the outgrowth of a project on the mathematical analysis of access delay in cable networks. As such, it is based on, and summarises, a number of publications, technical reports, patent filings, and two dissertations. The dissertations are Denteneer [47] and Van Leeuwaarden [109]. We now turn to a brief review of the publications and technical reports.

In Chap. 1, we have built upon the simulation approach described in Pronk and de Jong [144], Pronk et al. [145], and Hekstra-Nowacka et al. [78]. The observations on the finite-population effect stem from Denteneer and Pronk [55]. The material in Chap. 3 is based on Denteneer and Keane [54]. Chapter 4 is based on Boxma et al. [22, 23]. The sections on the gated service discipline have appeared earlier as Denteneer [44, 45]. Scheduled access has been extensively described in Denteneer [46]. The material in Chap. 5 is based on Denteneer and Gromoll [49].

Chapter 6, on the bulk service queue, is taken from Van Leeuwaarden [108]. Theorem 6.1 and the appendix of Chap. 6 are based on joint work with Adan and Winands, see [4]. The first modification of the classical bulk service queue, which couples the arrival process to the queue size, stems from Van Leeuwaarden et al. [111]. The model and the main results from Chap. 8 are taken from Denteneer and Van Leeuwaarden [52] and [50, 51, 53].

Chapters 9 and 10 are based on Van Leeuwaarden and Resing [112, 113]. The material in Chap. 11 stems from Denteneer [48].

<http://www.springer.com/978-3-540-69316-1>

Multiaccess, Reservations & Queues
Denteneer, D.; van Leeuwen, J.S.H.
2008, X, 254 p. 51 illus., Hardcover
ISBN: 978-3-540-69316-1