UNIVERSITEIT ANTWERPEN

Faculteit Wetenschappen

Departement Wiskunde-Informatica

Performance Evaluation of Contention Resolution Algorithms in Random Access Systems

Performantie Evaluatie van Conflict Resolutie Algorithmen in Random Access Systemen

Proefschrift voorgelegd tot het behalen van de graad van doctor in de Wetenschappen aan de

Universitaire Instelling Antwerpen te verdedigen door

Benny Van Houdt

Prof. Dr. Chris Blondia

Antwerpen, 2001

Introduction

This thesis focuses on the performance evaluation of a family of algorithms used to solve the so-called multiple access problem present in nearly all communication and computer networks. In order to define the multiple access problem consider two nodes part of a communication network. Such two nodes are connected with each other by a succession of communications links, the physical media of which can be coaxial cable, copper wire, fiber optics and radio spectrum. Broadly speaking, two types of communication network links exist. A point-to-point link consist of a single sender on one end of the link, and a single receiver at the other end of the link. The second type of link, a shared link, can have multiple sending and receiving nodes all connected to the same, single, shared link, e.g., wired and wireless local area networks (LANs), cellular access networks (GSM,GPRS), passive optical networks (PONs) and hybrid fiber coaxial networks (HFCs). Whenever a network solely consists of point-to-point links, there is no multiple access problem. However, if one or more shared links are present, a problem of central importance is how to coordinate the access of multiple sending and receiving nodes to a shared link. This problem is known as the multiple access problem. Protocols, or algorithms, designed to solve this problem are known as multiple access protocols.

An important subclass of multiple access protocols are so-called random access protocols (a definition is given in Chapter 1). The most commonly used random access protocols are the ALOHA protocols and the carrier sense multiple access (CSMA) protocols, e.g., Ethernet. Within this thesis we analyze the performance of another family of random access algorithms commonly known as tree algorithms and this both from a theoretical and a more practical point of view. Tree algorithms were developed during the late 1970s and since then a large body of literature has been devoted to them, especially during the 1980s. During the last five years they experienced yet another boost in attention with the development of hybrid fiber coaxial (HFC) and wireless (broadband) access networks.

Before we proceed with providing an overview of the contents of this thesis, it is useful to take a step back and first elaborate a bit about tree algorithms and their relation to the most important of all computer networks: the public Internet. We already indicated in the previous paragraph that tree algorithms received a lot of attention with the development of broadband access networks. Access networks are generally categorized into residential and company access networks. Nowadays, company access networks are completely dominated by Ethernet LANs. Until a few years ago, residential users were connected to the public Internet by means of a dialup modem over a POTS (plain old telephone system) or by means of an ISDN “telephone” line, which can be though of as a “better modem” [33] that supports rates up to 128 Kbps compared to the 56 Kbps dialup modems.

Two new technologies, asymmetric digital subscriber line (ADSL) and hybrid fiber coaxial cable (HFC) have been deployed during the last few years. ADSL runs over existing twisted-pair telephone lines and supports data rates between 2 and 8 Mbps from the Internet service provider (ISP) to a home. In the reverse direction the data rate is much smaller (between 16 and 640 Kbps). From the MAC perspective it is important to note that the uplink bandwidth, that is, from a home to the ISP, is not shared among different homes. HFC access networks differentiate themselves from ADSL, ISDN and dialup modems because they are an extension of the current cable networks used for broadcasting cable television. HFC access rates are comparable to ADSL, e.g., Motorola's CableCOMM system offers speeds downstream of up to 30 Mbps of which up to 10 Mbps is available to an individual modem and it runs smoothly upstream at a rate of up to 768 Kbps. However, with HFC, the upstream rates are shared among the homes. Therefore, a multiple access protocol is required. Due to the limited upstream bandwidth, upstream transmissions are reservation based, that is, a user has to reserve a part of the uplink bandwidth whenever it wants to transmit data. A mechanism, referred to as the access mechanism, that allows a user to reserve this bandwidth can be rather complicated [21, 35, 36]. However, a central feature of the access mechanism is a random access channel.

Formed in May 1994 by several vendors, the IEEE 802.14 Working Group (WG) develops international standards for data communications over cables, that is, HFC networks. Important for our discussion is that, after significant deliberations, the group selected a tree based algorithm for the random access channel [20, 21]. However, due to the delayed progress of the IEEE 802.14 WG, four major cable operators, Comcast Cable Communications, Cox Communications, Tele-Communications Inc., and Time Warner Cable, established the Multimedia Cable Network System (MCNS) Partners Ltd. in December 1995 to create the DOCSIS standard. The differences between the DOCSIS standard and the 802.14 draft were driven by organizational priorities. MCNS was aiming at keeping costs and market development to a minimum while IEEE was looking for a future-proof standard. The two standards differ the most in the medium access control (MAC) layer. Moreover, the DOCSIS standard replaced the tree algorithm by a simple binary exponential backoff (BEB) algorithm[1]. Extensive simulation studies, conducted by the National Institute of Standards and Technology (NIST), have indicated that the tree algorithm proposed by the IEEE 802.14 significantly outperforms the BEB algorithm in terms of delay and cell delay variation [20, 21]. Given these results the MCNS nevertheless selected the BEB algorithm for its simplicity. Knowing that "time is money" for the MCNS Partners, this came as no surprise.

DOCSIS v1.0 was approved as a standard by the ITU on March 19, 1998, and currently dominates the market. In addition, DOCSIS v1.1, whose major feature is supporting QoS service, was released on July 31, 1999. In contrast, the IEEE 802.14 Working Group was disbanded in March 2000, and IEEE 802.14a will remain as a draft afterward. The

[1]The BEB algorithm has been very successful in Ethernet LANs, however, the efficiency of Ethernet LANs is mainly guaranteed by the carrier sense and collision detection (CSMA/CD) mechanism combined with the limitations put on the length of a LAN segment. In HFC networks home users cannot sense nor detect collisions on the channel.

group has careful intentions and its specification is undoubtedly better than that developed by MCNS from a technological perspective [36]. Considering the European cable environment, the European Cable Communication Association (ECCA) started to create the EuroModem specification in December 1998. The EuroModem v1.0 was approved by the European Telecommunications Standard Institute (ETSI) on May 12, 1999. The contention resolution algorithm used in the EuroModem specification is the BEB algorithm.

Having discussed the relevance of tree algorithms in nowadays communication networks, we proceed with an overview of the contents of this thesis. The thesis is subdivided into two parts. The first analyzes the maximum stable throughput of tree algorithms, often referred to as their efficiency, under a number of idealized conditions. These conditions are used as the standard model of a multiple access link within the IEEE Information Theory Society [8]; hence, the multiple access problem is viewed from a theoretical perspective. A large body of papers has been written on this topic. Chapter 1 provides an overview of the most significant results and also includes a short discussion on other random access protocols not belonging to the class of tree algorithms. The main difference with all prior work is that we have significantly relaxed the assumptions made on the arrival process—an arrival process is a stochastic process that specifies how new packets are generated by the users (senders) connected to the shared link. Instead of Poisson arrivals we consider a rich class of tractable Markovian arrival processes, which lend themselves very well to modeling bursty arrival processes arising in computer and communication networks—namely, we consider discrete time batch Markovian arrival processes (D-BMAPs). Tree algorithms can be further categorized into three subclasses: the blocked access, free access and grouped access class. The methods used to analyze the first subclass—see Chapter 2—are fairly common and originated in the early 1980s [41]. To a certain extent the same can be said about the grouped access class (although some complications do arise, see Chapter 5). The free access class is by far the most difficult to analyze (given the current state of the art results) and requested a very different and new approach, Chapters 3 and 4 are devoted to them. The key result is to view a tree algorithm with free access as a tree structured quasi-birth-death (QBD) Markov chain, the theory of which was developed during the late 1990s, and to study the stability of the algorithm by means of the recurrence of the Markov chain. The main conclusion drawn from the first part of the thesis is that the good stability characteristics of tree algorithms under Poisson arrivals are maintained under this rich class of arrival processes, thereby further extending the established theoretical foundation of tree algorithms. More detailed conclusions and key results are found at the end of each chapter.

In the second part of the thesis, we study tree algorithms from a more practical perspective. Many access systems—for instance, wireless broadband systems, hybrid fiber coaxial (HFC) networks or passive optical networks (PONs)—have a point-to-multipoint architecture. The single end point, referred to as the access point (AP), operates as a centralized controller, that is, it decides which of the end nodes gets to transmit a packet to the AP. To make this decision, end nodes need to declare their bandwidth requirements to the access point (AP). This information is then used by the AP to schedule all uplink transmissions, that is, transmissions from an end node to the AP, according to the traffic characteristics and the quality of service (QoS) agreed upon. A problem of central importance is how the end nodes inform the AP about their bandwidth needs, a problem that has received considerable attention of the IEEE Communication Society. In the second part of this thesis, we address this problem in the context of wireless broadband access networks and we provide a detailed analysis of the *Identifier Splitting Algorithm combined with Polling (ISAP)*. The Identifier Splitting Algorithm is a tree algorithm that was introduced during the European RACE project 2067 on Mobile Broadband Systems (MBS). We have enhanced this algorithm with a polling mechanism and studied the influence of its parameters on the delay and throughput characteristics by means of several analytical models. These models combine elementary probability theory, queueing theory, combinatorics and the theory of Markov chains. The ISAP scheme is introduced in Chapter 6. Several analytical models that allow its evaluation are presented in Chapter 7, whereas in Chapter 8 we discuss the influence of the different protocol parameters by means of the analytical models presented in Chapter 7.

Before we proceed, there are a few people I would like to thank. First of all, I thank my promotor Chris Blondia for giving me the oppertunity to write a PhD (and to “act” like an assistant at the UIA for the past few years) and for introducing me to a number of mathematical approaches such as the matrix analytical methods. David Vázquez Cortizo for the useful discussions we had during his stay at the University of Antwerp. Moreover, I would like to express my gratitude to a number of international researchers for providing me with the necessary study material, especially P. Flajolet, J.M. Massey, M. Sidi, Q. He and M. Neuts. I also like to thank most of my colleagues at the University of Antwerp for the nice working atmosphere (in alphabetical order): David, Dennis, Floris, Hector, Joeri, Johan, Kathleen, Marc, Nico, Peter, Raf, Sandra, Stefan, Stijn, Tim, Toon and many more. Finally, I’m very grateful for the support I got from Lesley, my family and friends.

Contents

Part I

Stability of Tree Algorithms under Discrete time Batch Markovian Arrival (D-BMAP) traffic

Chapter 1

An Introduction to Random Access Algorithms

In this chapter we present a general introduction to random access algorithms. It is not our intention to provide a complete overview of all existing random access algorithms, nor to present them in a chronological order. Extensive overviews of random access algorithms can be found in [3, 63]. The emphasis of this introduction is on a family of random access algorithms commonly known as tree or splitting algorithms and on their stability characteristics. Before introducing the concept of a tree algorithm, we discuss the first, and one of the most popular, of all random access algorithms: the notorious ALOHA protocol. Some attention is also paid to acknowledgement-based, backoff and age-based algorithms. The chapter starts with a simple description of what a medium access control (MAC) protocol, or more specific a random access algorithm, is supposed to do.

1.1 Medium Access Control (MAC)

Broadly speaking, two types of network communication links exist. A point-to-point link consist of a single sender on one end of the link, and a single receiver at the other end of the link. The second type of link, a shared link, can have multiple sending and receiving nodes all connected to the same single, shared link. A shared link is often referred to as a shared medium. In the first scenario—that of the point-to-point link—there is no medium access control (MAC) layer present in the corresponding protocol stack. In the second scenario—that of the shared medium—multiple nodes might transmit simultaneously on the same link. A problem of central importance, to the data link layer, is how to coordinate the access of multiple sending and receiving nodes to a shared channel—the so-called multiple access problem. It is the task of the medium access control layer to regulate all transmissions on the shared link; i.e., to solve the multiple access problem. The Medium Access Control (MAC) sublayer is part of the data link layer in the ISO-OSI model [33, 63].

Since the early 1970s many MAC protocols have aroused. Most of them can be categorized as either being contention protocols or contention free protocols. Consider a shared link with a rate of R bits per second. In a contention protocol, or random access protocol, nodes always transmit at the full rate R of the link and are allowed to transmit simultaneously, although simultaneous transmissions seldom lead to a successful reception (the capture effect of a wireless channel is one of the few exceptions [49]). These simultaneous transmissions are referred to as collisions. Contention free protocols avoid collisions. There are two main protocol classes that avoid collision [33]. The first partitions the channel among all nodes sharing the link, e.g., time-division multiplexing (TDM), frequency-division multiplexing (FDM) or code-division multiplexing (CDM). The second class is known as the taking-turns protocols and allows nodes to use the channel during its turn, e.g., polling protocols and token-passing protocols. The main disadvantage of many contention free protocols is the low utilization of the network link. Both categories have proven their worth in a myriad of multiple access applications. More details are provided in the next few sections. In the remainder of this chapter an X channel refers to a channel upon which the MAC protocol X is being used (in literature the term ALOHA channel is sometimes also used for a channel that has certain characteristics).

Much attention has been paid to the stability of random access algorithms. A random access scheme is said to be stable if the mean time until a packet is transmitted successfully is finite. Underlying all the work done in this area are the following key assumptions [63]:

- New arrivals occur according to a Poisson process with rate λ.
- The number of nodes or stations is assumed to be infinite. In practice, the number of nodes is always finite. Assuming an infinite number provides us with an upper bound to the delay [3]. In particular, each finite set of nodes can regard itself as an infinite set of virtual stations, one for each arriving packet. This situation is equivalent to the infinite node assumption and allows a station with backlogged packets to compete with itself.
- A single error free contention channel provides immediate binary (collision or not) or ternary (collision, success or empty) feedback.

A lot can be—and has been—said about these assumptions and they are far from being the most realistic ones, but at least they provide us with a common framework in which we can make a fair comparison among different random access algorithms. When we discuss the stability of an algorithm under Poisson input traffic we actually refer to this common framework.

1.2 The ALOHA Protocols

This section is based on [3, 16, 41, 42, 63]. During the early days of communication networks (i.e., the old telephone networks) nodes were always connected using point-to-point connections. It was not until 1968, around the same period of time the first nodes of the ARPANET [32] were connected, that the first random access protocol, known as pure

ALOHA [1], came into existence. At the University of Hawaii researches were planning to interconnect a number of data terminals (stations), located on different isles, with the central computer by means of radio communication. The radio channel was to be shared among all stations. They proposed the following scheme to regulate all transmissions on the shared radio channel.

A station simply transmits whenever it has data to send. As stations send their frames at arbitrary times, there will be collisions. Frames involved in a collision are considered as destroyed and need to be retransmitted. The overlap between the colliding frames is irrelevant, namely, in all cases the checksum will fail and indicate that a retransmission is required. In order to reduce the number of collisions, stations retransmit a frame after a random delay between 0 and a predefined parameter δ. Stations that need to retransmit their frame are referred to as backlogged stations.

Assuming fixed length frames, pure ALOHA has a vulnerable period of 2 frames. Abramson indicated that the maximum throughput of a pure ALOHA channel (under Poisson traffic) is $1/2e$, i.e., about 18%, under what is called the *equilibrium hypothesis*. This hypothesis actually expresses the hope that the ALOHA channel is stable, i.e., that the mean waiting time of a packet is finite or in other words that the queue of frames awaiting retransmission is not growing steadily. As it turns out, ALOHA's simplicity causes it to be unstable for every arrival rate $\lambda > 0$ under Poisson input.

Roberts modified the ALOHA system by introducing the notion of "time slotting", this modified version is known as slotted ALOHA. Assuming fixed length frames, we choose this length as the unit of time. Stations are only allowed to start transmitting at a multiple of the time unit, thereby reducing the vulnerable period to a single frame and augmenting the maximum achievable throughput to $1/e$, i.e., about 36%, under the *equilibrium hypothesis*. Again, slotted ALOHA turned out to be unstable for all arrival rates $\lambda > 0$. There is also a geometric variant of Slotted ALOHA, where backlogged stations retransmit in each time slot with a probability p ($p = 1/\delta$). A simple proof that the geometric variant of the slotted ALOHA system is unstable for all arrival rates $\lambda > 0$ is given below.

In a slotted ALOHA system backlogged stations retransmit their frame in each slot with a probability p. Let a_i be the probability that i new arrivals occur in a slot. The number of new arrivals occurring in slot i and slot $i+1$ are independent and identically distributed. $N(t)$, the number of backlogged stations during time slot t, is therefore a Markov chain on the state space $\{n \mid n \geq 0\}$ with the following transition probabilities $P_{k,l}$

$$\begin{aligned}
P_{k,k-1} &= a_0 kp(1-p)^{k-1}, \\
P_{k,k} &= a_0(1-kp(1-p)^{k-1}) + a_1(1-p)^k, \\
P_{k,k+1} &= a_1(1-(1-p)^k), \\
P_{k,k+j} &= a_j \ (j \geq 2).
\end{aligned}$$

For $0 < p < 1$ and $a_0 + a_1 < 1$, $N(t)$ is an aperiodic irreducible Markov chain and slotted ALOHA is stable if and only if this Markov chain is ergodic. Obviously, for k large enough, $P_{k,k-1} < 1 - a_0 - a_1 = \sum_{j\geq 2} P_{k,k+j}$ because $P_{k,k-1}$ decreases to zero. Moreover, $P_{k,k-i} = 0$

for $i > 1$. Therefore, the Markov chain $N(t)$ does not have a stationary distribution as a result of the Instability Lemma by Kaplan [3, p265]. This is sufficient to prove slotted ALOHA's instability for every arrival process with $a_0 + a_1 < 1$, in particular for the Poisson arrival process with a mean $\lambda > 0$ ($a_i = \lambda^i/i!\ e^{-\lambda}$).

Kelly further improved this result by showing that the number of successful transmissions on an AHOLA channel is finite with probability 1. In conclusion, eventually an ALOHA channel becomes jammed with collisions. The time that elapses before this occurs can however be very large. For instance, Greenberg and Weiss have shown that for $p = 0.01$ and $\lambda = 0.1$ it takes about e^{346} time slots before the channel is "jammed" with collisions. Numerous proposals have been made to stabilize ALOHA, each one proposing a different method on how to estimate the number of backlogged stations. None of them succeed in keeping the virtue of the original ALOHA schemes: their simplicity.

ALOHA systems are nevertheless often implemented in practice, although most of them appear in fact to be unstable. In order to cope with the instability, they implement some kind of "time out" feature that clears the system if totally jammed with collisions. This solution works fine when the traffic intensity—that is, the rate of the new arrivals—and the retransmission probability p is low.

1.3 Acknowledgement-based, Backoff and Age-based Algorithms

Another important random access scheme, known as Ethernet, was introduced in 1979 by Metcalfe (Harvard) [45]. Stations making use of an Ethernet channel postpone the i-th retransmission attempt for a random time between 0 and 2^i time units, as opposed to 0 and δ on an ALOHA channel. Ten years after the introduction of Ethernet, Aldous (Berkeley) [2] proved that Ethernet was unstable for all arrival rates $\lambda > 0$ under Poisson arrivals. The instability of Ethernet is not as severe as that of ALOHA. For instance, Kelly and MacPhee [30] have shown that the number of successful transmissions is finite, resp. infinite, with probability 1 if $\lambda > \ln 2 = .69$, resp. $\lambda < \ln 2 = .69$, for the slotted version of Ethernet. Whereas the number of successful transmissions on an ALOHA channel is finite with probability 1 for all $\lambda > 0$. In practice, Ethernet frames are dropped if the number of retransmission attempts reaches a predefined threshold. ALOHA and Ethernet both belong to a class of algorithms known as acknowledgement-based algorithms[1]. In an acknowledgement-based algorithm, users make retransmission decisions using only the history of their own transmission attempts—that is, users only receive feedback from the channel indicating whether their own transmission attempts are successful or not. Other algorithms that listen to the feedback of every slot are referred to as full-sensing algorithms (examples are the tree algorithms presented in the next section). Recently, Goldberg *et al* [19] have shown that all acknowledgement-based algorithms are unstable

[1]Notes on contention resolution written by L.A. Goldberg from the Warwick University were very useful in writing the remainder of this section. The notes are unpublished and a copy can be found at her webpage: http://www.dcs.warwick.ac.uk/~leslie.

for $\lambda > .530045$ under Poisson input. Moreover, not even one acknowledgement-based algorithm is known so far to be provably stable for any arrival rate $\lambda > 0$ under Poisson traffic.

An important subclass of the acknowledment-based algorithms are the backoff algorithms. A backoff algorithm is associated with a sequence of probabilities p_i, $i \geq 0$. In a given time slot of the corresponding algorithm, every station that has a packet ready for transmission and that has been unsuccessful in transmitting this packet on i occasions transmits (independently) with probability p_i. Obviously, slotted ALOHA and Ethernet are backoff algorithms with $p_i = p$ and $p_i = 2^{-i}$ respectively. It has been shown that backoff algorithms are always unstable under Poisson traffic for $\lambda \geq .42$ [19]. Tree algorithms are therefore superior to backoff algorithms—from the stability point of view—because there are many tree algorithms known that support higher input rates (up to .48776, see Section 1.4). In 1989 MacPhee posed the question whether there exists a backoff algorithm that is stable for any $\lambda > 0$. The answer to this question is still unknown. Kelly, *et al* [30] have shown that all backoff algorithms with slower than exponential backoff result in a finite number of successful transmissions with probability 1. For instance, setting $p_i = (i+1)^{-k}, k \geq 1$, results in a finite number of successful transmissions (with probability 1).

Another interesting subclass are the age-based algorithms. An age-based algorithm is associated with a sequence of probabilities $p_i, i \geq 0$. In a given time slot of the corresponding algorithm, every station (re)transmits (independently) with probability p_i if the packet was generated i time slots ago. Kelly and MacPhee have shown that the number of successful transmissions is finite if and only if $\sum_{i=1}^{t} p_i$, i.e., the expected number of transmissions that a packet endures in the first t slots after being generated, is $\Omega(\log(t))^2$ (^{2} this is a footnote mark). Ingenoso has shown that age-based algorithms are unstable if $p_i, i \geq 0$, is monotonically decreasing. For instance, setting $p_i = a/i$ results in an infinite number of successes because $\sum_{i=1}^{t} 1/i = \log t + O(1)$, but the algorithm is nevertheless unstable.

1.4 Tree Algorithms

The breakthrough in searching for a random access scheme that was provably stable was made by Capetanakis [7] in 1977 and independently by Tsybakov and Mikhailov [64] and to some extent by Hayes [3]. The basic idea behind this scheme was already used by Dorfman during the Second World War for testing soldiers for syphilis [12, 63] and is an algorithm for what is known as the group testing problem. The group testing problem studies algorithms to find d defects in a population of size N as fast as possible. A single test on a group of n indicates whether there is at least one defect in the group of size n. For instance, the syphilis soldiers are the defects among all soldiers. Dorfman used the following method: take a blood sample from N soldiers and mix a portion of each sample into a single sample. Next, test this sample for syphilis. If negative, all soldiers are

[2] A function $f(t) = \Omega(g(t))$ if $\forall c > 0\ \exists N : f(t) \geq cg(t)$ for $t \geq N$.

cleared. Otherwise create two samples: one by mixing a portion of the first N/2 soldiers together, the second using the portions of the last N/2 soldiers. This algorithm is applied recursively until the identity of all the syphilis soldiers is known. Whether this algorithm minimizes the number of tests required, for $d > 1$, is still an open issue. For $d = 1$ it is proven to be the fastest possible.

When translated to a computer network this algorithm goes as follows: whenever a group of n stations collides, they split into 2 groups. Each station draws a pseudo random number to decide whether it joins the first or the second group. Stations joining the first group retransmit in the next slot and resolve a possible collision recursively, while the other stations wait until the first group is resolved before applying the same algorithm. A station joins the first, resp. the second, group with probability p_1, resp. $p_2 = 1 - p_1$. Whenever a station selects a group it is said to flip a coin. For $p_1 = 1/2$ the coin is said to be fair, otherwise it is referred to as biased. The collision resolution algorithm (CRA) described above is known as the basic binary Capetanakis-Tsybakov-Mikhailov (CTM) or tree algorithm. It can be combined with different channel access protocols (CAPs). A channel access protocol indicates when a newly arrived packet is allowed to transmit for the first time.

For now we discuss the following two CAPs:

- Blocked Access: After an initial collision of n stations, all new arrivals postpone their first transmission attempt until the n initial stations have resolved their collision. The time elapsed from the initial collision until the point where the n stations have transmitted successfully is called the collision resolution interval (CRI). Suppose that m new packets are generated during the CRI. Then, a new CRI starts (with m participants) when the previous CRI (with n stations involved) ends. In conclusion, when the blocked access mode is used new arrivals are blocked until the CRI during which they arrived has ended. They will participate in the next CRI.

- Free Access: New arrivals transmit the moment they are generated, i.e., at the first slot boundary following their arrival time. Thus, if k new arrivals occur during slot i and the n stations that transmitted in slot i split into a group of n_1 and n_2 stations, $n_1 + k$ stations will transmit in slot $i + 1$.

Different terminology is used when these channel access protocols (CAPs) are combined with a tree algorithm. For instance, the blocked access schemes are also referred to as tree-search algorithms, the free access schemes as stack algorithms [35]. Implementation details and examples are provided in Chapter 3. Binary feedback (collision or not) suffices in order to implement the basic binary tree algorithm with blocked or free access. Many other tree algorithms have aroused from this initial one. An overview is presented Sections 1.4.1 to 1.4.5.

An important result for the Poisson input traffic that applies to any random access scheme implementing a blocked access strategy is the following [22, 41]. If a conflict resolution algorithm (CRA) has an expected running time $T(n)$, to resolve n participants, then the corresponding random access algorithm with blocked access is stable for all $\lambda <$

$\liminf n/T(n)$ and is unstable for $\lambda > \limsup n/T(n)$. The expression for $T(n)$ depends upon the conflict resolution algorithm. Whatever happens for an arrival rate λ between the liminf and the limsup of $n/T(n)$ is unclear (although in some particular cases some light was shed on this gray area, see [43]). For some CRAs $n/T(n)$ does have a limit[3] for $n \to \infty$, i.e., the gray area disappears, but this is not always the case[4] (although the size of the gray area tends to be rather small in such cases). In Chapter 2 we will generalize this result to a more general class of arrival processes.

The key result in studying the stability of the basic CTM algorithm with blocked access was, strangely enough, already obtained by Knuth in 1973 [17]. The reason is the generality of the recursive process based on random choices that turns out to be the exact model for a variety of searching algorithms in computer science. Let l_N denote the expected number of slots required to solve a collision of N stations. Knuth showed that l_N satisfies the following equation asymptotically (for $p_1 = 1/2$, i.e., fair coins):

$$l_N = \frac{2}{\ln 2} N + N P(\log_2 N) + O(\sqrt{N}), \tag{1.1}$$

with $P(\cdot)$ a periodic function with an amplitude $< 10^{-5}$. Combining this result with the property mentioned in the previous paragraph, shows that the CTM algorithm with blocked access (and fair coins) is stable for $\lambda < \ln 2/2 - 10^{-5}$ and unstable for $\lambda > \ln 2/2 + 10^{-5}$ under Poisson traffic. In Chapter 3, we prove that this result is not merely valid for the Poisson arrival process. Knuth's result was however not commonly known at the time. For instance, in 1981 Massey [41] showed that the CTM algorithm with blocked access was stable under Poisson input for $\lambda < .3465$ and unstable for $\lambda > .3471$. In this paper Massey mentions that W. Sandrin of the Comsat Laboratories pointed out that $\ln 2/2 \approx .3465735$. In 1985 Mathys and Flajolet [13, 43] showed that the best stability results for the Poisson input traffic are obtained with fair coins, i.e., $p_1 = 1/2$.

In general, studying the stability of a random access scheme with free access is more difficult compared to a blocked access scheme. In 1985 Mathys and Flajolet [13, 43] eventually showed that the basic binary CTM algorithm with free access, also referred to as the binary stack algorithm, is stable under Poisson input traffic (and fair coins) for $\lambda < .360$. Moreover, for the Poisson traffic fair coins are the optimal coins; that is, they achieve the highest maximum stable throughput. In Chapter 3, we show that both these results are not valid for other arrival processes. Chapter 3 presents analytical methods that allow us, among other things, to determine the stability of the basic binary CTM algorithm with free access for a variety of arrival processes.

[3]For instance, when slotted ALOHA is combined with blocked access, it is easy to show $\lim_{n\to\infty} n/T(n) < \lim_{n\to\infty} n^2 p \; e^{(n-1)\ln(1-p)} = 0$. Therefore, slotted ALOHA with blocked access is unstable for all arrival rates $\lambda > 0$, whatever the value of the retransmission probability p.

[4]In 1980 Vvedenskaya [41] was the first to prove that $\lim_{n\to\infty} n/T(n)$ does not exist for many tree algorithms. However, a lot of the Russian results were unknown to the Western world for quite some time.

1.4.1 The Basic Q-ary CTM or Tree Algorithm

A first of many generalizations of the basic binary tree algorithm is the basic Q-ary tree algorithm. This generalization consists of splitting the set of stations involved in a collision into Q—instead of two—groups. Stations part of the i-th group postpone any retransmission attempts until the first $i-1$ groups have been resolved. A station selects the i-th group with a probability p_i. Whenever $p_1 = p_2 = \ldots = p_Q = 1/Q$ one talks about fair coins, otherwise about biased coins. For the new packet arrivals one can either use free or blocked access. The stability properties of the basic Q-ary tree algorithm were revealed by Mathys and Flajolet [43] in 1985 and can be summarized as follows.

We start with the basic Q-ary tree algorithm with blocked access. Let l_N denote the expected number of slots required to solve a collision of N stations. Then, l_N/N satisfies the following equation asymptotically:

$$l_N/N = \frac{Q}{-\sum_{i=1}^{Q} p_i \ln p_i} + f_1(N) + O(N^{-1}), \tag{1.2}$$

with $f_1(N)$ a fluctuating function of small amplitude, between 10^{-3} and 10^{-6}. Due to the property of any blocked access algorithm we find that the basic Q-ary tree algorithm is stable for $\lambda < -\sum p_i \ln p_i/Q - \epsilon$ and unstable $\lambda > -\sum p_i \ln p_i/Q + \epsilon$, for some ϵ small, under Poisson input traffic. The sum $-\sum_{i=1}^{Q} p_i \ln p_i$ reaches a maximum equal to $\ln Q/Q$ for $p_i = 1/Q$, $1 \leq i \leq Q$. Therefore, the basic Q-ary tree algorithm (with fair coins) is stable for arrival rates up to $\lambda \approx \ln Q/Q$. The highest arrival rates (up to .3662) can be supported by the ternary scheme, i.e., $Q = 3$, followed by the binary and quaternary schemes who both support rates up to .3466. For $Q = 5$ we get .3218 and the maximum achievable throughput $\ln Q/Q$ further decreases for higher splitting factors Q (see Table 1.1 and Figure 1.1).

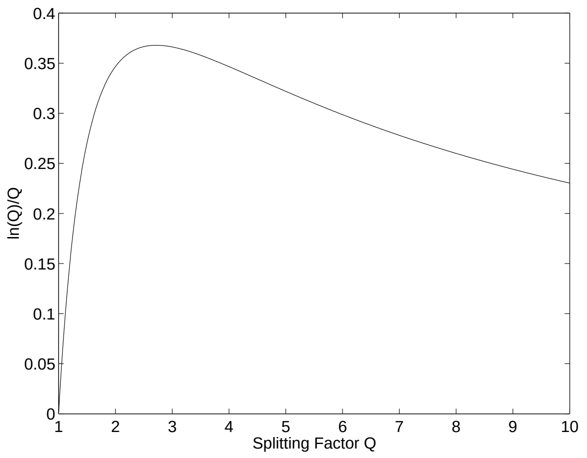


Figure 1.1: *Influence of the Splitting Factor on the Maximum Stable Throughput for the Basic Q-ary CTM Algorithm*

As noted before, the maximum achievable throughput of a random access scheme with blocked access (under Poisson traffic) is found by studying the asymptotic behavior of

Q	*basic blocked access*	*basic free access*	*mod. blocked access*	*mod. free access*
2	.3466	.3602	$\underline{.3754}$	.3872
3	$\underline{.3662}$	$\underline{.4016}$	.3741	$\underline{.4070}$
4	.3466	.3992	.3496	.4007
5	.3219	.3872	.3233	.3878
6	.2986	.3734	.2994	.3736
7	.2780	.3597	.2784	.3598
8	.2600	.3470	.2602	.3471
9	.2441	.3353	.2443	.3353
10	.2303	.3246	.2304	.3246

Table 1.1: *Maximum achievable throughput for the basic and the modified Q-ary tree algorithm with fair coins*

$n/T(n)$, where $T(n)$ is the expected runtime of the conflict resolution algorithm (CRA) required to resolve the contention between n participants. This runtime $T(n)$ is also referred to as the expected length of a collision resolution interval (CRI) initiated by n participants. For free access algorithms a CRI is generally defined as the time that elapses between two successive time instances for which none of the stations has a packet ready for transmission. A random access algorithm with free access is stable whenever the expected length of an arbitrary CRI is finite, otherwise it is unstable (an asymptotic analysis of the length of a CRI is not required). The results for the basic Q-ary CTM algorithm with free access are presented in Table 1.1. Fair coins achieve the best stability results.

1.4.2 The Modified Q-ary CTM or Tree Algorithm

The basic Q-ary tree algorithm exploits binary feedback (collision or not). It can be improved by exploiting ternary (collision, success or empty) feedback whenever available [3, 41]. The algorithm that exploits ternary feedback is referred to as the modified Q-ary tree algorithm. It can be combined with both blocked and free access and works as follows. If, after a collision, the next $Q-1$ slots turn out to be empty—that is, all stations involved in the collision chose the last group and no new arrivals occurred if the free access strategy is used—the next slot must contain a collision if the basic Q-ary CTM algorithm is used as the conflict resolution algorithm. This otherwise *doomed slot* can be skipped by having all stations act as if the collision had occurred. Obviously, the modified scheme performs at least as well as the basic algorithm. Surprisingly, Capetanakis failed to notice the existence of certain-to-occur collisions in his algorithm. Massey [41] was the first to point this out, whereas Tsybakov and Mikhailov discovered this independently; as a consequence the modified CTM algorithm is also referred to as the CMTM (Cap-Mas-Tsy-Mik) algorithm.

In practice, this improvement has a slight problem, when combined with the blocked access strategy, in that if an idle slot is incorrectly perceived by the receiver as a collision—this might happen in an environment in which errors occur—the algorithm continues splitting indefinitely. Let us explain this phenomenon. Suppose that an empty slot is perceived as a collision due to an error in the channel. As a result all stations, including those generating new arrivals, wait until the set of stations involved in this collision is resolved, but this set is an empty set. Therefore, the next $Q-1$ slots are empty (new arrivals are blocked) and the modified algorithm kicks in and skips the slot following these $Q-1$ empty slots (because it believes that this slot necessarily contains a collision). Notice, if the basic algorithm were to be used the next slot would have turned out empty and the "collision" would have been resolved. As for the modified algorithm, the next $Q-1$ slots are again empty and another slot will be skipped by the modified scheme. As a result the algorithm becomes deadlocked as it continues splitting indefinitely; that is, none of the stations ever succeed in transmitting their packet. In practice, after some predefined number h times $Q-1$ empty slots, where every $Q-1$ slots are followed by a split, the algorithm should allow the next subset to transmit without first splitting it. The value of h depends upon the reliability of the medium.

The stability characteristics of the modified algorithm under Poisson traffic were also revealed by Mathys and Flajolet [43]. The corresponding equation for the blocked access scheme (with fair coins) for l_N/N is

$$l_N/N = \frac{Q(1-f_1(N)) - [Q^{-1} + (1-Q^{-1})\ln(1-Q^{-1}) - f_2(N)]}{\ln Q} + O(N^{-1}), \quad (1.3)$$

were $f_1(N)$ and $f_2(N)$ are fluctuating functions of small amplitude, between 10^{-3} and 10^{-6}. Numerical values for $Q=2$ to 10 are found in Table 1.1. This table also represents the results for the free access scheme. Fair coins are, for both the blocked access and free access strategy, no longer the optimal coins. It turns out that increasing the probability p_Q, while keeping the others equal to each other, slightly improves the maximum achievable throughput. For instance, the modified ternary tree algorithm with free access supports input rates up to .407614 for $p_1 = p_2 = .314544$ and $p_3 = .370911$. The modified binary tree algorithm with blocked access achieves a stability of .381260 for $p_1 = .4174$ and $p_2 = .5826$.

1.4.3 Estimating the Multiplicity of Conflicts to Speed Their Resolution

The highest stability result under Poisson traffic we encountered, so far, when exploiting binary, resp. ternary, feedback is .401599, resp. .407614. Higher stability results, up to .487 for ternary feedback, have been achieved in a variety of ways. The first, discussed in this section, can be used in combination with a blocked access strategy and exists in estimating the number of participants at the start of the collision resolution interval (CRI). If the estimated multiplicity is equal to m, all stations taking part in the CRI split into m groups at the start of the CRI. Next, each of the m groups is resolved using a collision resolution algorithm (in our case a tree algorithm). This idea was first introduced by Capetanakis in his dynamic tree protocol, under the assumption that the multiplicity of the conflict was a Poisson distributed random variable [3, 22]. Several procedures have

a	*basic binary tree*	*modified binary tree*	*modified biased binary tree*
2	.4025	.4341	.4402
1.1	.4202	.4526	.4589
1.01	.4256	.4581	.4644
1.001	.4275	.4602	.4665
1.0001	.4282	.4609	.4672

Table 1.2: *Maximum achievable throughput for the basic, modified and the biased modified* ($p_1 = .4174$) *binary tree algorithm when combined with the base a estimation method*

been proposed for estimating the conflict multiplicity. A summary of those whose accuracy does not depend on the stochastic assumptions about the arrival process is presented in this section.

Greenberg, *et al* [22] proposed the following estimation method known as the base a estimation algorithm. The base a estimation algorithm searches for a power of a that is close to n, the conflict multiplicity. The following probabilistic test of the hypothesis that $n \geq a^i$ is used. Let each of the n conflicting stations transmit in a slot with probability a^{-i}. A collision supports the hypothesis that $n \geq a^i$. This test is executed with $i = 1, 2, 3, \ldots$ until no collision occurs. If this procedure leads to a series of j collisions, n is estimated as a^j. The estimation therefore requires $1 + \log_a n^* = O(\log_a n)$ time slots, where n^* is the estimate for n [22]. The closer we choose a to one, the better the estimate turns out to be.

Greenberg, *et al* [22] determined the asymptotic behavior of the expected time l_N required to resolve a CRI with N participants when the basic binary tree algorithm, the modified binary tree algorithm and the modified biased binary tree algorithm ($p_1 = .4174$) is used as the contention resolution algorithm. Combining this with the Poisson property for blocked access schemes provides us with the maximum achievable throughput. Numerical results are presented in Table 1.2. The results indicate that stability up to .4282, resp. .4672, can be achieved by exploiting binary, resp. ternary, feedback.

Cidon and Sidi [8] further experimented with the estimation ideas of Greenberg *et al* [22]. They proposed the following estimation procedure. Suppose that there are n contenders in the CRI. Each of the n stations transmits in the first slot of the CRI with a probability $p > 0$. Thus, the n stations are split into two sets E and D, where E consists of those stations that transmitted and D of the others. If this first slot holds a collision—that is, $|E| \geq 2$—then the stations in E use the modified binary CTM or tree algorithm to resolve the collision. When the set E is resolved we know the number of participants $|E|$ in E. The estimate for n, denoted as n^*, is computed as $|E|/p$ and the estimate for $|D|$ is $n^* - |E|$. Next, m is defined as $\max\{1, \lceil \alpha(n^* - |E|) - \beta \rceil\}$. The parameter β has no effect on the stability of the algorithm, whereas α is used to optimize the stability. Next, the stations belonging to the set D are split into m sets and each set is resolved using the modified binary CTM algorithm. Cidon and Sidi [8] have shown that the $\liminf n/T(n)$ of this conflict resolution algorithm is equal to .468 for $\alpha = .786$ and $p < 10^{-5}$. Using this idea they constructed a more complex variation on this conflict resolution algorithm and found one for which $\liminf n/T(n) = .487$—that is, a CRA that resolves conflicts of multiplicity n, for n large, in expected time of approximately $2.054n$ time slots.

1.4.4 Grouping on Arrival Times

Another natural way to devise a random access algorithm that achieves a high stable throughput is to "decouple" transmission times from arrival times[5]. This was first suggested by Gallager [3] and his Russian counterpart Federov [41]. A description by Massey [41] is given below. Suppose that the random access scheme is activated at time $t = 0$. The unit of time is defined as the length of a slot, so that the i-th transmission slot is the time interval $(i, i + 1]$. A second time increment Δ is chosen and the i-th arrival epoch is defined as the time interval $(i\Delta, i\Delta + \Delta]$ (Δ is not necessarily an integer value). The first transmission rule used by this algorithm is as follows: transmit a new packet that arrived during the i-th arrival epoch in the first "utilizable" slot following the collision resolution interval (CRI) for new packets that arrived during the $(i-1)$-th arrival epoch. The modifier "utilizable" reflects the fact that the CRI for new packets that arrived during the $(i-1)$-th arrival epoch might end before the i-th arrival epoch has ended. If so, a number of transmission slots are skipped until the i-th arrival epoch ends. One could improve the algorithm by shortening the i-th arrival epoch. This both complicates the analysis and the implementation and has no influence on the maximum stable throughput.

Each of the groups is resolved using either the basic binary or the modified binary tree algorithm, depending on whether we have binary or ternary feedback (the order in which they are resolved is of no importance). Conflict resolution algorithms that use a higher splitting factor ($Q > 2$) are not considered for resolving the groups. The reason is the following. When grouping arrivals based on the arrival epochs, it is important to have a collision resolution algorithm that performs well for groups with very few contenders (because these appear the most frequent if Δ is small). The basic Q-ary tree algorithm performs best in resolving groups with $n \leq 3$ participants for $Q = 2$. The same can be said about the modified algorithm for $n \leq 7$. This causes higher splitting factors to achieve worse stability results (if Δ is small).

Massey [41] has proven that the maximum stable throughput achieved by this algorithm under Poisson input is .4294, resp. .4622, when exploiting binary, resp. ternary, feedback by setting Δ equal to 2.6712, resp. 2.7066. Notice, the expected number of arrivals in an arrival epoch is 1.147, resp. 1.251. Gallager [3] further improved this algorithm by making the result of the coin flip depend upon the arrival times. Thus, packets generated during the first half of the interval, which is being resolved, are considered as flipping "0", the others as flipping "1". An important consequence is that the resulting algorithm is a first-come-first-served (FCFS) algorithm, namely, the order in which the stations are successful is identical to the order of arrival. Gallager also indicated a second improvement that increases the maximum achievable throughput and greatly simplifies the analysis. Consider what happens when a collision is followed by another collision in

[5]The idea of grouping has been reintroduced more recently in wireless local area networks (LANs) with delayed feedback [9, 10]

the tree algorithm. Let n be the number of stations involved in the first collision and assume that n_1 of the n stations select the first group; thus, $n_2 = n - n_1$ select the second. Let n_{11}, resp. n_{12}, be the number of stations that select the first, resp. second, group after the second collision. The second collision indicates that $n_1 \geq 2$. Due to the first collision we have $n = n_1 + n_2 \geq 2$, therefore we know nothing about n_2 (for Poisson arrivals). Gallager therefore suggested to add the n_2 stations to the group of the n_{12} stations. The analysis of this algorithm is much easier compared to other tree algorithms because the status of all backlogged stations (those who do not belong to the set that is currently being resolved) is identical. Gallager proved that this algorithm, referred to as the FCFS splitting algorithm, supports rates up to .4871 (when exploiting ternary feedback). Mosely and Humblet [28] further refined the algorithm for rates up to .48776.

1.4.5 The Deterministic Tree Algorithm

Capetanakis, Hayes, and Tsybakov and Mikhailov [22] independently proposed a deterministic tree algorithm. A deterministic tree algorithm is used as a random access algorithm in an environment with a finite set of K stations. Each of these K stations is identified by a unique number, written as a Q-ary number and referred to as the MAC address of the station (K is chosen as a power of Q). It differs from the basic Q-ary tree algorithm with blocked access in the sense that stations are no longer split into Q sets using a probabilistic method, but use their MAC address in a deterministic fashion. A station selects the i-th group after the j-th collision in a CRI if the j-th digit of its MAC address equals i. As a result the maximum length of a CRI is reduced to $(QK-1)/(Q-1)$, corresponding to a full Q-ary tree of height k, where $K = Q^k$, that develops when all K stations are active. The maximum delay for a message is therefore bounded by $2(QK-1)/(Q-1) < 4K$.

The conditions under which the stability analysis of an algorithm operating in a finite population of K stations is done, are very different from those operating in an infinite population. For a finite population one assumes that each station generates traffic according to a Poisson process with rate λ/K, resulting in a global rate λ [17]. A station attempts to transmit one packet at a time, while the other packets are buffered until a successful transmission takes place. The algorithm is said to be stable if the expected delay of a packet is finite. Obviously, stability is maintained for all rates $\lambda < (Q-1)K/(QK-1)$. $(Q-1)K/(QK-1)$ decreases as K increases and reaches a limit of $(Q-1)/Q$ for K to infinity.

1.5 Upper Bounds on the Maximum Achievable Throughput

This section is based on [38, 41, 42]. Much work has gone into determining upper bounds on the maximum achievable throughput that can be supported by a full-sensing algorithm under Poisson input traffic. In a full-sensing algorithm, all users receive feedback information at the end of each slot (as opposed to the acknowledgement-based algorithms). Pipperger was the first to improve the obvious upper bounds[6] using information-theoretic arguments. He showed that all algorithms are unstable for $\lambda > .744$. Humblet improved this bound in 1979 to .704. The next improvement was made by Molle in 1980: all algorithms are unstable for $\lambda > .6731$. Kelly (1985) introduced a new boundary for algorithms that allow new arrivals to transmit immediately. The argument went as follows. Suppose that an algorithm is operating stable under Poisson arrivals. Define p_r as the fraction of the slots in which retransmissions take place. Then, because the number of new senders in any slot is independent of the number of retransmitted packets in that slot, it follows that the fraction of slots with exactly one packet (i.e., the throughput τ) satisfies

$$\tau \leq \lambda e^{-\lambda}(1 - p_r) + p_r e^{-\lambda} \tag{1.4}$$

with equality when at most 1 packet is retransmitted in a slot. The stability implies $\tau = \lambda$. It is readily[7] seen that setting $p_r = 1$ maximizes λ for which the equation can be satisfied; thus, $\lambda \leq e^{-\lambda}$. The largest λ that satisfies this equation is $\lambda = .5671$. Notice, even if the stations were to communicate among each other in order to implement a collision free retransmission scheme, but do not exchange information about the new arrivals which are transmitted immediately, one cannot achieve a throughput above .5671. Tsybakov and Mikhailov (1987) used similar but more intricate arguments to prove that all random access algorithms are unstable for $\lambda > .5683$. The best algorithms known achieve a throughput of .4492, resp. .4877, when binary, resp. ternary, feedback is exploited.

In 1979 Mosely gave some convincing arguments (but no proof) that random access schemes for which the order of the successful transmissions is identical to the order of the arrivals, i.e., FCFS algorithms, cannot achieve a throughput above .48785. Recently, in 1998, Loher has proven that FCFS algorithms are always unstable for $\lambda > .4906$. The FCFS splitting algorithms of Gallager and Mosely are stable for rates up to .4871 and .4877.

[6]It is easy to show that random access algorithms that allow new arrivals to transmit immediately are unstable under Poisson traffic for $\lambda/(1+\lambda) < e^{-\lambda}$, i.e., $\lambda > .802$.

[7]Equation (1.4) can be rewritten as $g(\lambda) = (e^{\lambda} - 1)\lambda/(1-\lambda) \leq p_r$, with $g(0) = 0, g(\lambda) \geq 0$ for $0 \leq \lambda < 1$ and $g'(\lambda) > 0$ for $0 \leq \lambda < 1$. As a result one maximizes the solution by setting $p_r = 1$.

Chapter 2

D-BMAPs and Random Access Algorithms with Blocked Access

The objective of this chapter is threefold. First, we describe a class of arrival processes commonly known as discrete time batch Markovian arrival processes (D-BMAPs), discuss some of its properties and present some examples. Second, we motivate why it is useful to study the stability of random access schemes under D-BMAP input traffic. Finally, we prove that the Poisson results presented in Chapter 1 for the random access algorithms with blocked access are also valid for most D-BMAPs. Blocked access is one of the channel access protocols (CAPs) presented in Chapter 1. The other two, namely, free access and grouping, are discussed in Chapters 3–5. That is, the stability of tree algorithms with free access under D-BMAP input traffic is addressed in Chapter 3 and 4; while tree algorithms that make use of grouping (see Section 1.4.4) are discussed in Chapter 5.

2.1 D-BMAPs: Definition, Properties and Examples

The D-BMAP is the discrete time counterpart of the BMAP [39, 40] and was first introduced in [4]. D-BMAPs form a class of tractable Markovian arrival processes, which, in general, are non-renewal and which include the discrete time variants of the Markov modulated Poisson process, the PH-renewal process and superpositions of such processes as particular cases. Because of its versatility, it lends itself very well to modeling bursty arrival processes commonly arising in computer and communications applications [5, 47, 48].

2.1.1 A Definition

A definition by Daniëls [11] is given below. Formally, a D-BMAP is defined by an infinite set of positive $l \times l$ matrices $(B_n)_{0 \leq n < \infty}$, with the property that

$$B = \sum_{n=0}^{\infty} B_n \tag{2.1}$$

is a transition matrix. A D-BMAP is denoted by $(B_n)_n$, which completely determines it. By definition the Markov chain J_t associated with B and having $\{i; 1 \leq i \leq l\}$ as its state space, is controlling the actual arrival process as follows. Suppose J is in state i at time t. By going to the next time instance $t+1$, there occurs a transition to another or possibly the same state, and a batch arrival may or may not occur. The entries $(B_n)_{i,j}$ represent the probability of having a transition from state i to j and a batch arrival of size n. So, a transition from state i to j without an arrival will occur with probability $(B_0)_{i,j}$. Define by X_t the number of arrivals generated at time t.

D-BMAPs are generally defined with l, the size of the square matrices B_n, finite. It is possible to extend their theory for l infinite [11]. However, the D-BMAPs studied in this thesis are assumed to have a finite number of states. Some of the properties we prove with respect to random access schemes, make explicit use of the finiteness of l. We also assume that the transition matrix B is an aperiodic irreducible matrix. Aperiodic irreducible matrices are often referred to as primitive matrices [58]. Thus, whenever we refer to a primitive D-BMAP we mean to say that its transition matrix B is aperiodic and irreducible. For B primitive the Markov chain J_t has a unique stationary distribution. Let β be the stationary probability vector of the Markov chain J_t, i.e., $\beta B = \beta$ and $\beta e = 1$ with e a column vector of 1's. The mean arrival rate $\lambda = E[X_t]$ of the D-BMAP $(B_n)_n$ is given by

$$\lambda = \beta \left(\sum_{n=1}^{\infty} n B_n \right) e. \tag{2.2}$$

Due to the Ergodic Theorem for primitive Markov Chains [58] we have

$$\lim_{L \to \infty} \frac{E[\sum_{i=0}^{L-1} X_{t+i} \mid J_t = j]}{L} = \lambda, \tag{2.3}$$

for $1 \leq j \leq l$. D-BMAPs for which $B_n = 0$, for $n \geq 2$, are referred to as discrete time Markovian arrival processes (D-MAPs).

2.1.2 Some Properties

The following properties have been shown to hold for an arbitrary D-BMAP $(B_n)_n$. Additional properties and discussions can be found in [4–6]. First, a superposition of two D-BMAPs $(B_n^1)_n$ and $(B_n^2)_n$ is again a D-BMAP $(B_n)_n$. The B_n matrices of the newly created D-BMAP are calculated as a sum of Kroncker products between the B_n^1 and B_n^2 matrices, see [4, 11]. Second, the autocorrelation function $r(k) = \text{Cov}(X_1, X_k)/\text{Var}(X_1)$ is found as [4]

$$r(k) = \frac{\beta \left[\sum_{n=1}^{\infty} n B_n\right] B^{k-2} \left[\sum_{n=1}^{\infty} n B_n\right] e - \lambda^2}{\beta \left[\sum_{n=1}^{\infty} n^2 B_n\right] e - \lambda^2}. \tag{2.4}$$

The index of dispersion for counts (IDC), a measure for the burstiness of an arrival process, at time k, is defined as

$$I(k) = \frac{\text{Var}(\sum_{j=1}^{k} X_j)}{E[\sum_{j=1}^{k} X_j]} = \frac{k\text{Cov}(X_1, X_1) + 2 \sum_{j=1}^{k-1} (k-j) \text{Cov}(X_1, X_{j+1})}{\lambda k}. \tag{2.5}$$

Another measure that is often used for the burstiness is the index of dispersion for intervals (IDI). The IDI is the sequence c_k^2 defined as

$$c_k^2 = \frac{kVar[\sum_{j=1}^k S_j]}{E[\sum_{j=1}^k S_j]^2}, \tag{2.6}$$

where S_j represents the j-th interarrival time. For a renewal process [61] we have $c_k^2 = c_1^2$, where c_1^2 is the squared coefficient of variation, i.e., the variation divided by the square of the mean, of the number of arrivals in a slot. In particular, for the Poisson process $I(k) = c_k^2 = 1$.

2.1.3 Some Examples

The Discrete Time Poisson Process

The discrete time Poisson process is obtained by observing the continuous time Poisson process at the slot boundaries. Arrivals that occurred in the interval $(t, t+1]$ are now assumed to arrive on the boundary of slot t and $t+1$, i.e., at time $t+1$. We can model the discrete time Poisson process as a D-BMAP with a single state by letting $B_n = e^{-\lambda}\lambda^n/n!$, for $n \geq 0$. The autocorrelation function $r(k) = 0$, while the index of dispersion for counts (IDC) $I(k) = 1$. Whether we use the continuous time or discrete time variant of the Poisson process makes no difference to the stability of a time slotted algorithm. The mean delay is slightly different (at most 1). For later reference, we abbreviate the Poisson process as $PP(\lambda)$.

The Discrete Time Erlang Process

We define the continuous time Erlang process as follows. The continuous time Erlang process has independent and identically distributed interarrival times that obey an Erlang distribution [23] with parameters k and λ_e (this λ_e is not to be confused with the arrival rate λ of the corresponding D-BMAP). Clearly, for $k = 1$ the Erlang process is reduced to the Poisson process. By observing the Erlang process at the slot boundaries we obtain the discrete time Erlang process (arrivals are assumed to occur on slot boundaries). The discrete time Erlang process can be modeled as a D-BMAP in the following way. Let $\gamma_n = e^{-\lambda}\lambda^n/n!, n \geq 0$, and let $B_n, n \geq 0$, be $k \times k$ matrices defined as

$$(B_n)_{i,j} = \gamma_{nk+j-i} \qquad nk \geq j - i, \tag{2.7}$$

$$(B_n)_{i,j} = 0 \qquad nk < j - i. \tag{2.8}$$

The arrival rate λ of this D-BMAP[1] is λ_e/k. For later reference, we abbreviate the Erlang k process as $ER(\lambda_e, k)$.

[1]The matrix $B = \sum_n B_n$ is a circulant matrix [60]. Therefore, it is possible to determine the eigenvalues of B explicitly as a function of λ_e and k. Which allows us to get an explicit expression for the decay rate of the autocorrelation function [11].

The Discrete Time Markov Modulated Poisson Process

We restrict ourselves to the discrete time Markov modulated Poisson processes with two states. These processes are characterized by two parameters λ_1, λ_2 and a 2×2 matrix T. The process will generate arrivals according to a Poisson process with a mean rate λ_i when the current state is i. Transitions from one state to another can occur at the end of each time slot according to a 2×2 transition matrix T

$$T = \begin{pmatrix} 1 - \frac{1}{a} & \frac{1}{a} \\ \frac{1}{b} & 1 - \frac{1}{b} \end{pmatrix}. \tag{2.9}$$

The expected sojourn time in state 1, resp. state 2, is a, resp. b, time slots. The matrices B_n are found as

$$B_n = \begin{pmatrix} \frac{\lambda_1^n e^{-\lambda_1}}{n!}(1 - \frac{1}{a}) & \frac{\lambda_1^n e^{-\lambda_1}}{n!}\frac{1}{a} \\ \frac{\lambda_2^n e^{-\lambda_2}}{n!}\frac{1}{b} & \frac{\lambda_2^n e^{-\lambda_2}}{n!}(1 - \frac{1}{b}) \end{pmatrix}. \tag{2.10}$$

Notice, $\sum_n B_n = B = T$. The arrival rate λ is calculated as $(\lambda_1 a + \lambda_2 b)/(a + b)$. By means of the spectral decomposition [11] of T and Equation (2.4) it is not too difficult to find the autocorrelation function $r(k)$

$$r(k) = \frac{(1 - \frac{1}{a} - \frac{1}{b})^{k-1}(\lambda_1 - \lambda_2)^2}{\frac{\lambda_1 a}{b} + \frac{\lambda_2 b}{a} + \lambda_1 + \lambda_2 + (\lambda_1 - \lambda_2)^2}. \tag{2.11}$$

For later reference, we abbreviate the Markov Modulated Poisson process with parameters λ_1, λ_2, a and b as $M(\lambda_1, \lambda_2, a, b)$.

The Bulk Arrival Process

The Bulk arrival process is defined as a discrete time arrival process characterized by a $1 \times m$ vector v and a length L. The arrival pattern of this process consists of a repetition of identical cycles. The first part of each cycle consists of a set of batches, characterized by v. For instance $v = [2, 3, 2]$ means that we first have a batch of size 2, in the next time slot we have a batch of size 3, followed by a batch of size 2. The second part of the cycle is a silent period with a geometrically distributed length with average L. The Bulk arrival process can be described by the following D-BMAP. Let $v = [v_1, \ldots, v_m]$ and let $B_n, n \geq 0$, be a set of $(m + 1) \times (m + 1)$ matrices with

$$(B_{v_j})_{j,j+1} = 1 \; (1 \leq j \leq m), \tag{2.12}$$

$$(B_0)_{m+1,1} = 1/L, \tag{2.13}$$

$$(B_0)_{m+1,m+1} = 1 - 1/L. \tag{2.14}$$

The other components of the matrices B_n are equal to zero. The arrival rate λ of a Bulk arrival process equals $\sum_j v_j/(L + m)$. For $m = 1$ one easily obtains that the autocorrelation function $r(k)$ obeys the following equation:

$$r(k) = -\frac{(-1)^k L^{-(k-2)}}{L - 2}. \tag{2.15}$$

For later reference, we abbreviate the Bulk arrival process with parameters v and L as $B(v, L)$.

2.2 D-BMAPs as Access Network Input Traffic

It has been pointed out in literature [8, 22, 42] that the stability of a random access algorithm under a more general class of arrival processes—also referred to as the robustness of an algorithm [8, 42] or, equivalently, the insensitivity to the statistics of the arrival process—is an attractive practical feature. The reason is obvious: in practice, an access network, e.g., a local area network (LAN), operates with a finite number of users and traffic generated on such a network tends to be more bursty and correlated compared to Poisson arrivals. The class of D-BMAPs allows us to incorporate burstiness and correlation and is therefore, to some extent, better suited to match access network input traffic. As with the Poisson arrivals we assume that the D-BMAP traffic is generated by an infinite number of users, this provides us with an upper bound to the delay.

The fact that we limit ourselves to discrete time arrival processes is of no importance. The stability under a continuous time batch Markovian arrival process can be studied by creating a discrete time variant with the same stability properties. The discrete time variant is found by observing the continuous time process at the slot boundaries and by assuming that the arrivals that occurred in the interval $(t, t+1]$ actually occur at time $t+1$, i.e., on the boundary of time slot t and $t+1$ (e.g., the discrete time Poisson and Erlang processes described in Section 2.1.3). Notice, the time interval $(t, t+1]$ is referred to as slot t. Suppose that the D-BMAP $(B_n)_n$ is used as input traffic and assume that the D-BMAP is in some state i, $1 \leq i \leq l$, at time t. Then, with a probability $(B_n)_{i,j}$, the state at time $t+1$ is j and n new packets are generated at the boundary of slot $t-1$ and t. In a random access algorithm with free access these n new packets are transmitted—for the first time—in time slot t by their corresponding stations. In a blocked access scheme each of these n stations defers the first transmission attempt until the current collision resolution interval (CRI) has finished. Whereas in a grouping algorithm, they postpone the transmission attempt until all prior groups have been resolved (unless the groups are not resolved in a FCFS order).

2.3 D-BMAPs and Blocked Access Algorithms

Recall from Chapter 1, that if the input traffic is Poisson with a mean λ and if a conflict resolution algorithm (CRA) has an expected running time $T(n)$, to resolve n participants, then the corresponding random access algorithm with blocked access is stable for all $\lambda < \liminf n/T(n)$; unstable for $\lambda > \limsup n/T(n)$. The expression for $T(n)$ depends upon the CRA. Therefore, it is sufficient to study the asymptotic behaviour of $n/T(n)$ for n to infinity in order to determine the stability of a blocked access scheme under Poisson input. This behaviour is, obviously, independent of the arrival process. Thus, in order to generalize the stability results of any blocked access scheme, presented in Chapter 1, it suffices to generalize the above-mentioned Poisson property to the arrival process of interest.

Comments that this property can be generalized to other arrival processes are often found in literature [14, 22, 42, 43]. For instance, Massey [42] states that "This stability holds not only for the assumed Poisson arrival process, but for *virtually* any arrival process that can be characterized by an average arrival rate λ." Massey [41] proves the property for Poisson arrivals and gives an intuitive argument for other arrival processes. Cidon and Sidi [8, Theorem 8] have proven the following theorem. Let $\sigma = \liminf n/T(n)$ and let $N_{t,t+L}$ be the number of packets arriving to the system in the interval $(t, t+L]$. Then, the system is stable if there exists a $\delta > 0$ and an L^* such that $E[N_{t,t+L}] < (\sigma - \delta)L$ for all t and $L > L^*$. For instance, assuming Poisson arrivals, $E[N_{t,t+L}]$ is nothing but λL for all t . Hence, by setting $L^* = 1$, it suffices to find a $\delta > 0$ such that $\lambda < (\sigma - \delta)$, where $\sigma = \liminf n/T(n)$. In conclusion, we have stability if $\lambda < \liminf n/T(n)$.

From Section 2.1.1 we know that the expected number of arrivals of a primitive D-BMAP that occur in an interval of length L approaches λL as L approaches infinity, where λ is the mean arrival rate (whichever the state at the start of the interval is). Provided that the number of states of the D-BMAP l is finite, we find that for any $\epsilon > 0$ there exists an L^* such that $E[N_{t,t+L}] < (\lambda + \epsilon)L$ for all t and $L > L^*$. Thus, when $\lambda + \epsilon < \sigma$, it suffices to choose δ equal to $(\sigma - \lambda) - \epsilon > 0$ to fulfill the required equation (ϵ is chosen to be smaller than $\sigma - \lambda$). Hence, we have the following theorem:

Theorem 2.1 *A random access algorithm with blocked access, corresponding to a conflict resolution algorithm (CRA) that resolves conflicts of multiplicity n in expected time $T(n)$, is stable under primitive D-BMAP $(B_n)_n$ input traffic if*

1. *$\lambda < \liminf n/T(n)$, with λ the mean arrival rate,*
2. *$(B_n)_n$ has a finite number of states l.*

The aperiodicity of the D-BMAP $(B_n)_n$ is not really a requirement, i.e., the theorem is also valid under irreducible D-BMAP traffic. We did not find a proof in the literature showing that the system becomes unstable for $\lambda > \limsup n/T(n)$ (except for Poisson arrivals). Therefore, we now prove the following theorem. The proof method is a generalization of Massey's proof for the Poisson arrivals [41].

Theorem 2.2 *A random access algorithm with blocked access, which corresponds to a collision resolution algorithm (CRA) that solves collisions of multiplicity n in an expected time $T(n)$, is unstable under primitive D-BMAP traffic if*

1. *$\lambda > \limsup n/T(n)$, with λ the mean arrival rate,*
2. *$(B_n)_n$ has a finite number of states l,*
3. *$(B_n)_n$ is not a D-MAP, that is there exists a $n > 1$ such that $B_n \neq 0$.*

We start with the following definitions. Let $(B_n)_n$ be a primitive D-BMAP with a finite number of states, i.e., with l finite. Let Y_i and X_i denote the length and the number of participants of the i-th collision resolution interval (CRI), where X_0 and Y_0 correspond to the CRI beginning at time $t = 0$. Let Z_i denote the state of the D-BMAP at the start of the i-th CRI, where Z_0 is the state at time $t = 0$. Let $T(n)$ be the expected time required by the conflict resolution algorithm (CRA) to resolve a set of n contenders, i.e., $T(n) = E[Y_i \mid X_i = n]$. Using the law of total probability, we have

$$E[Y_i] = \sum_{n=0}^{\infty} P[X_i = n] E[Y_i \mid X_i = n]. \tag{2.16}$$

Let $\tau = \limsup n/T(n)$, then for any $\epsilon_1 > 0$ there exists an $N(\epsilon_1)$ such that $n/T(n) \leq \tau + \epsilon_1$ for $n > N(\epsilon_1)$. In other words, $T(n) \geq n/(\tau + \epsilon_1)$ for $n > N(\epsilon_1)$. Therefore, we can write Equation (2.16) as

$$E[Y_i] \geq \frac{1}{\tau + \epsilon_1} \sum_{n > N(\epsilon 1)} n P[X_i = n] + \sum_{n \leq N(\epsilon_1)} E[Y_i \mid X_i = n] P[X_i = n]. \tag{2.17}$$

Let $T(n) = n/(\tau + \epsilon_1) + g(n)$, where $g(n)$ is a correction that can be either positive or negative. Therefore,

$$E[Y_i] \geq \frac{1}{\tau + \epsilon_1} E[X_i] + \sum_{n \leq N(\epsilon_1)} g(n) P[X_i = n]. \tag{2.18}$$

Whenever $g(n) \geq 0$ we use 0 as a lower bound for $g(n)P[X_i = n]$; otherwise, we use $g(n)$ as an lower bound for $g(n)P[X_i = n]$. Hence,

$$E[Y_i] \geq \frac{1}{\tau + \epsilon_1} E[X_i] + e(\epsilon_1), \tag{2.19}$$

where $\epsilon_1 > 0$, $e(\epsilon_1) \leq 0$ is a fixed number[2] that does not depend upon i and $\tau = \limsup n/T(n)$. We know from Section 2.1.1 that for any primitive D-BMAP the expected number of arrivals in an interval of length L approaches λL as L approaches infinity, where λ is the arrival rate of the D-BMAP (independent of the state at the start of the interval). Thus, because the number of states of the D-BMAP $(B_n)_n$ is finite, we have that for any $\epsilon_2 > 0$ there exists a $K(\epsilon_2)$ such that $E[X_{i+1} \mid Y_i = L] \geq (\lambda - \epsilon_2)L$ for $L > K(\epsilon_2)$. Hence, by means of the law of total probability

$$E[X_{i+1}] \geq (\lambda - \epsilon_2) \sum_{L > K(\epsilon_2)} L P[Y_i = L] + \sum_{L \leq K(\epsilon_2)} P[Y_i = L] E[X_{i+1} \mid Y_i = L]. \tag{2.20}$$

Recall Z_i is the state of the D-BMAP at the start of the i-th CRI. Obviously,

$$E[X_{i+1} \mid Y_i = L] \geq \min_j E[X_{i+1} \mid Y_i = L \cap Z_i = j]. \tag{2.21}$$

The expression $\min_j E[X_{i+1} \mid Y_i = L \cap Z_i = j]$ is nothing but the expected number of arrivals generated by the input D-BMAP during an interval of length L, provided that the state at the start of the interval is j. Hence, we can write it as $(\lambda - \epsilon_2)L + h(L)$, where $h(L)$ is a correction that is either positive or negative, to obtain

$$E[X_{i+1}] \geq (\lambda - \epsilon_2) E[Y_i] + \sum_{L \leq K(\epsilon_2)} h(L) P[Y_i = L]. \tag{2.22}$$

[2]The value $e(\epsilon_1)$ also depends on the CRA being used and not solely on ϵ_1.

For $h(L)$ negative, resp. positive, we replace $h(L)P[Y_i = L]$ by $h(L)$, resp. 0, to find that

$$E[X_{i+1}] \geq (\lambda - \epsilon_2)E[Y_i] + f(\epsilon_2), \tag{2.23}$$

where $f(\epsilon_2) \leq 0$ is a fixed number[3] that does not depend upon i. Combining Equations (2.19) and (2.23) provides us with the following equation:

$$E[X_{i+1}] \geq \frac{\lambda - \epsilon_2}{\tau + \epsilon_1} E[X_i] + (\lambda - \epsilon_2)e(\epsilon_1) + f(\epsilon_2), \tag{2.24}$$

for $i \geq 0$. When the equality is taken in Equation (2.24), we have a first-order linear recursion whose solution for the initial condition $X_0 = N$ and $Z_0 = j$ is a lower bound on $E[X_i]$. This lower bound can be rearranged to the following form:

$$E[X_i] \geq \left(N - \frac{[(\lambda - \epsilon_2)e(\epsilon_1) + f(\epsilon_2)]}{1 - \frac{\lambda - \epsilon_2}{\tau + \epsilon_1}}\right)\left(\frac{\lambda - \epsilon_2}{\tau + \epsilon_1}\right)^i + \frac{[(\lambda - \epsilon_2)e(\epsilon_1) + f(\epsilon_2)]}{1 - \frac{\lambda - \epsilon_2}{\tau + \epsilon_1}}, \tag{2.25}$$

with $e(\epsilon_1) \leq 0$ and $f(\epsilon_2) \leq 0$. Define $\frac{[(\lambda - \epsilon_2)e(\epsilon_1) + f(\epsilon_2)]}{1 - (\lambda - \epsilon_2)/(\tau + \epsilon_1)}$ as I_N[4]. For $(\lambda - \epsilon_2) > (\tau + \epsilon_1)$ we find $I_N \geq 0$ (ϵ_2 is chosen such that $\lambda > \epsilon_2$). Thus, for $(\lambda - \epsilon_2) > (\tau + \epsilon_1)$ the lower bound for $E[X_i]$ presented in Equation (2.25) grows without a bound as i goes to infinity if N is large enough—that is, larger than I_N. For N smaller than I_N the lower bound for $E[X_i]$ decreases to minus infinity and we know nothing from Equation (2.25).

Notice, Equation (2.25) actually states that if a CRI with more than I_N participants occurs, $E[X_i]$ grows without bound—that is, the algorithm is unstable under D-BMAP $(B_n)_n$ traffic—for $\lambda > \tau$. Next, we prove that a CRI with more than I_N contenders occurs with probability one if $(B_n)_n$ is not a D-MAP. Consider the Markov chain (X_i, Z_i) on the state space $\{(n,j) \mid n \geq 0, 1 \leq j \leq l\}$. From this Markov chain, we construct a finite state Markov chain W_i with an absorbing state w by replacing the states $\{(n,j) \mid n > I_N, 1 \leq j \leq l\}$ by a single absorbing state w. The state space of W_i is $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$, with $\Omega_1 = \{(n,j) \mid n = 0 \text{ or } 1, 1 \leq j \leq l\}$, $\Omega_2 = \{(n,j) \mid 2 \leq n \leq I_N, 1 \leq j \leq l\}$ and $\Omega_3 = \{w\}$. Hence, Ω consists of $(I_N + 1)l + 1$ states. Denote the transition matrix P of W_i as

$$P = \begin{pmatrix} A & B & a \\ C & D & b \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.26}$$

where the $2l \times 2l$ matrix A, resp. $2l \times (I_N - 1)l$ matrix B, represents the transition probabilities from the states in Ω_1 to those in Ω_1, resp. Ω_2. Whereas, the $(I_N - 1)l \times 2l$ matrix C, resp. $(I_N - 1)l \times (I_N - 1)l$ matrix D, represents the transition probabilities from the states in Ω_2 to those in Ω_1, resp. Ω_2. Finally, let the vectors a, resp. b, represent the

[3]Obviously, $f(\epsilon_2)$ also depends on the D-BMAP $(B_n)_n$ being used as the input process and on the conflict resolution algorithm (CRA) being used because $K(\epsilon_2)$ depends on ϵ_2 and the CRA.

[4]Notice, the value of I_N depends upon ϵ_1, ϵ_2, the collision resolution algorithm (CRA) that is used (because t, $e(\epsilon_1)$ and $f(\epsilon_2)$ depend on the CRA) and the D-BMAP (because λ and $f(\epsilon_2)$ depend on the D-BMAP).

probabilities that a transition is made from the states in Ω_1, resp. Ω_2, to the absorbing state w. Moreover, denote the k-th power of P as

$$P^k = \begin{pmatrix} A^{(k)} & B^{(k)} & a^{(k)} \\ C^{(k)} & D^{(k)} & b^{(k)} \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.27}$$

The states in $\Omega_1 \cup \Omega_2$ are transient [31] if for some $k' > 0$: $a^{(k)} > 0$ and $b^{(k)} > 0$ for $k > k'$. In which case a transition to the absorbing state eventually occurs with probability one. That is, a CRI with more than I_N contenders occurs with probability one.

First, we have $b > 0$; hence, $b^{(k)} > 0$ for any $k > 0$. Indeed, if a CRI has $n \geq 2$ contenders, then for each $t > 0$ there exists a non zero probability $p_{n,t}$ that the CRA needs t or more time slots to resolve the contention between n participants (because the CRA works with an infinite population). Also, for t large enough, there exists a non zero probability that more than I_N arrivals occur in an interval of length t (whatever the state at the start of the interval). As a result, there exists a non zero probability that a CRI with two or more contenders is followed by a CRI with more than I_N contenders. Second, in order to show that there exists a $k' > 0$ such that $a^{(k)} > 0$ for $k > k'$, it suffices to show that each of the $2l$ rows of the $2l \times (I_N - 1)l + 1$ matrix $[B^{(k)} a^{(k)}]$ with $k \geq k'$ has at least one entry that defers from zero (because $b > 0$). Suppose that the Markov chain (X_i, Z_i) is in state $(n_1, j_1) \in \Omega_1$. The input D-BMAP $(B_n)_n$ is not a D-MAP; hence, there exists an i^*, j^* and $m > 1$ such that $(B_m)_{i^*,j^*} \neq 0$. Moreover, due to the irreducibility of $B = \sum_n B_n$ we know that for some $k_{(n_1,j_1)} > 0$ there exists a non zero probability that the Markov chain (X_i, Z_i) makes a $k_{(n_1,j_1)}$-step transition from state (n_1, j_1) to a state of the form (n_2, i^*)— that is, a state of the form (n_2, i^*) is reached after $k_{(n_1,j_1)}$ transitions with a non zero probability. From the state (n_2, i^*) there is a non zero probability (because $(B_m)_{i^*,j^*} \neq 0$) that the Markov chain (X_i, Z_i) makes a transition to a state of the form (n_3, j_2) with $n_3 \geq m > 1$. Hence, from the state (n_1, j_1) there is a non zero probability that a state of the form (n_3, j_2) is reached in $k_{(n_1,j_1)} + 1$ steps, with $n_3 \geq m > 1$. Thus, if the Markov chain W_i is in the state $(n_1, j_1) \in \Omega_1$, there exists a nonzero probability that W_i reaches a state in $\Omega_2 \cap \Omega_3$ after $k_{(n_1,j_1)} + 1$ steps. Therefore, choosing $k' = \max_{(n_1,j_1)\in\Omega_1} k_{(n_1,j_1)} + 1$ completes the proof.

In conclusion, the algorithm corresponding to the conflict resolution algorithm that solves conflicts of multiplicity n in an expected time $T(n)$, is unstable under the D-BMAP $(B_n)_n$ if $\lambda > \limsup n/T(n)$ and if $(B_n)_n$ does not belong to the class of D-MAPs. As far as the D-MAPs are concerned, Equation (2.25) also states that the algorithm is unstable for D-MAPs with $\lambda > \limsup n/T(n)$ if the number of participants in the first CRI is sufficiently large. This may seem somewhat counter intuitive. For instance, for each CRI there exists a non zero probability that no new arrivals occur during the CRI (except for $\lambda > 1$ and some periodic D-MAPs). If this happens we obviously get stability because all subsequent CRIs have either zero or one participants. Notice, Equation (2.25) states that such an event does not happen with a probability one. In conclusion, starting with a CRI with more than I_N participants, with D-MAP input traffic with an arrival rate $\lambda > \limsup n/T(n)$, results in an unstable algorithm because the expected delay is infinite. Often there is however a non zero probability that stability is obtained along the way (this probability should be equal to one in order to obtain a finite expected delay).

2.4 Conclusion

In this chapter we introduced the class of D-BMAP arrival processes and motivated why it is useful to study the stability of a random access algorithm under D-BMAP input traffic. We also demonstrated that it is fairly easy to prove that the stability/unstability of a blocked access algorithm under primitive D-BMAP traffic (with a finite number of states and not belonging to the class of D-MAPs) is identical to the stability/unstability under Poisson traffic. This is a very positive characteristic of a blocked access scheme. Obviously, this does not imply that the delay is in the same order of magnitude for different arrival processes. The objective of Part I of this thesis is to study the stability of most of the tree algorithms presented in Section 1.4, with the exception of some of the grouping algorithms of Section 1.4.4, under D-BMAP input traffic. Having dealt with the blocked access schemes, the basic and modified Q-ary CTM algorithms with free access and a number of grouping algorithms remain to be studied. Indeed, the estimation algorithms presented in Section 1.4.3 are also of the blocked access type. In the next chapter we introduce a method to study the stability of the basic binary CTM algorithm with free access under D-BMAP input traffic. In Chapter 4 we generalize this method to the basic and modified Q-ary CTM algorithm with free access; whereas Chapter 5 deals with tree algorithms that make use of a grouping strategy.

Chapter 3

Analysis of the Basic Binary CTM Algorithm with Free Access

In this chapter we indicate how to determine whether the basic binary CTM algorithm with free access is stable under D-BMAP $(B_n)_n$ input traffic. We start with a more detailed description of the algorithm to be studied in order to get a good grasp of the problem and how the mathematics relate to the problem. Afterwards, we introduce a class of Markov chains known as Quasi-Birth-Death (QBD) Markov chains with a tree structure and indicate how to construct such a Markov chain that is recurrent, resp. transient, whenever the basic binary CTM algorithm (with free access) is stable, resp. unstable. An algorithm that determines whether this Markov chain is recurrent or not is also provided. Furthermore, using this Markov chain we can calculate the mean delay experienced by a packet and many other performance measures of interest. Numerical results are presented at the end of the chapter. It is important to notice that to our best knowledge tree structured Markov chains have never been used in order to study a medium access control protocol. So far, applications of tree structured Markov chains have been limited to the study of last come first serve (LCFS) queueing systems with multiple classes of customers, each class having a different service requirement [24–26, 62, 78, 79]. The work presented in this chapter was published in [68].

3.1 The Basic Binary CTM Algorithm with Free Access

From Chapter 1 we know that the basic binary CTM algorithm is a collision resolution algorithm (CRA) for which each user strives to retransmit its colliding packet till it is correctly received. The users have to resolve this contention without the benefit of any additional information on other users' activity. The algorithm separates, in a recursive way, users that collide into two groups. The separation is done according to some randomization procedure. The users that select the first group attempt a retransmission in the next slot, while the users that select the second group wait until the first group is resolved.

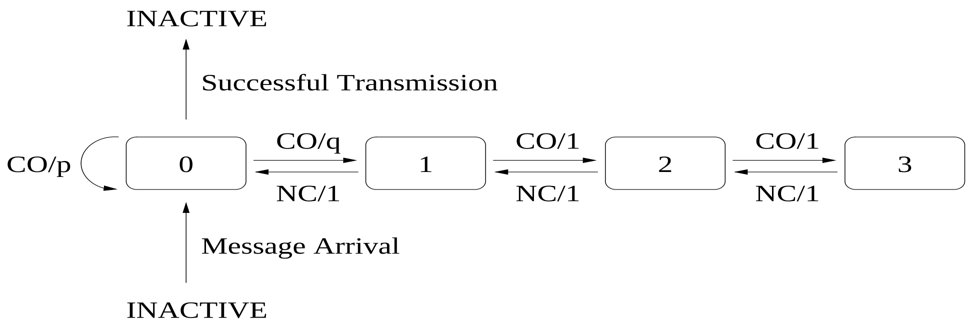


Figure 3.1: *State Diagram: CO = collision, NC = no collision*

In correspondence with the framework used to study the stability under Poisson input (see Chapter 1), we have an infinite number of users, i.e., stations, each holding zero or one packets. Users that hold a packet (at time t) are referred to as active users (at time t). The basic binary CTM protocol is conveniently implemented by letting each active user maintain an integer value, referred to as the current stack level. The current stack level held by a station can be seen as a representation of the number of "groups" that need to be resolved before a station is allowed to (re)transmit. At the end of each time slot the current stack level is updated according to the following rules (see Figure 3.1):

- A user that became active, i.e., generated a new packet, during slot $t-1$ initializes its current stack level for slot t at zero. A user is allowed to transmit in time slot t whenever its current stack level for slot t is zero. Therefore, users that became active during slot $t-1$ transmit in slot t (together with other stations that have their current stack level for slot t at zero).

- Suppose that slot t does not hold a collision, i.e., at most one user has its current stack level for slot t at zero. Then, users with a current stack level for slot t equal to $i, i > 0$, set their current stack level for slot $t+1$ at $i-1$ (while a possibly successful user becomes inactive).

- If slot t however does hold a collision, users with a current stack level for slot t equal to $i, i > 0$, set their current stack level for slot $t+1$ at $i+1$. While, users with a current stack level for slot t equal to zero split into two groups: a user joins the first group with a probability p and the second group with a probability $q = 1-p$. All the users that join the first group set their current stack level for slot $t+1$ at zero, while the users that join the second group set their current stack level for slot $t+1$ at one.

An example of the transmission process is shown in Figure 3.2. Figure 3.2 also includes a list of group numbers (1 or 2) for each packet to indicate which group the packet joins after each collision (in which it is involved). Thus, the list $1, 2, \ldots$ for packet E indicates that packet E joins the first group as a result of its first collision and the second as a result of its second collision.

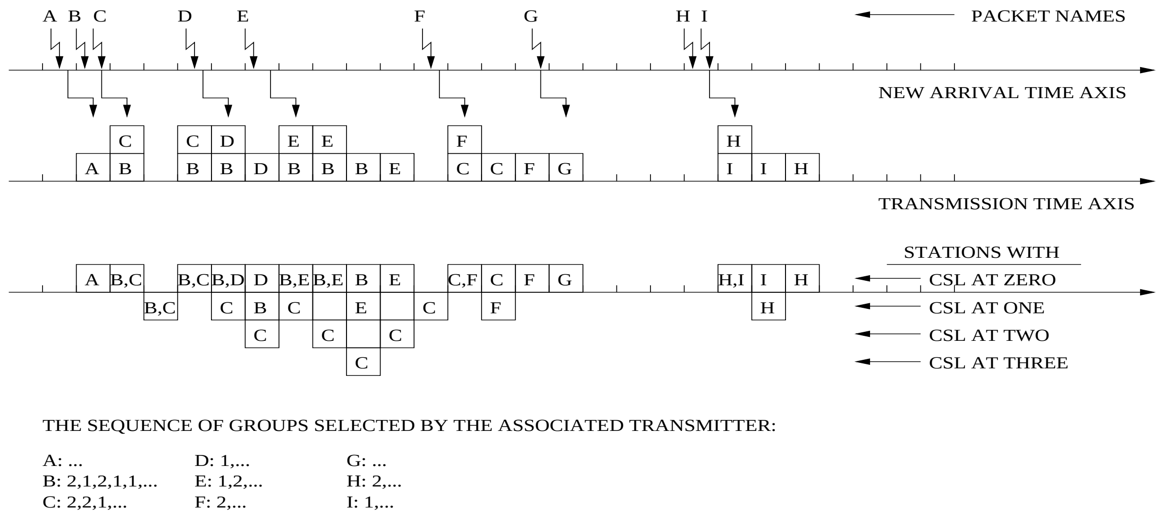


Figure 3.2: *Example of the Transmission Process: CSL = Current Stack Level*

3.2 Markov Chain of the Quasi-Birth-Death Type with a Tree Structure

In this section we briefly describe a tree structured Quasi-Birth-Death (QBD) Markov chain. This type of Markov chain was first introduced in Takine, *et al* [62] and Yeung, *et al* [78, 79]. The theory of tree structured QBD Markov chains is a generalization of the well-known theory of matrix analytical methods introduced by Neuts [47, 48]. The generalization exists in considering the discrete time bivariate Markov chain $\{(X_t, N_t), t \geq 0\}$, in which the values of X_t are represented by nodes of a Q-ary tree, and where N_t takes integer values between 1 and m. X_t is referred to as the node and N_t as the auxiliary variable of the Markov chain at time t. A description of the transitions of the Markov chain is given below. A Q-ary tree is a tree for which each node has Q children. The root node is denoted as $\emptyset$. The remaining nodes are denoted as strings of integers, with each integer between 1 and Q. For instance, the k-th child of the root node is represented by k, the l-th child of the node k is represented by kl, and so on. Throughout this chapter we use lower case letters to represent integers and upper case letters to represent strings of integers when referring to nodes of the tree. We use '+' to denote concatenation on the right. For example, if $J = 1\ 0\ 8, k = 6$ then $J + k = 1\ 0\ 8\ 6$.

The Markov chain (X_t, N_t) is called a Markov chain of the QBD-type with a tree structure if at each step the chain can only make transitions to its parent, children of its parent (including itself), or to its children, see Figure 3.3. Moreover, if the chain is in state $(J + k, i)$ at time t then the state at time $t + 1$ is determined as follows:

1. (J, j) with probability $d_k^{i,j}, k = 1, \ldots, Q$,
2. $(J + s, j)$ with probability $a_{k,s}^{i,j}, k, s = 1, \ldots, Q$,
3. $(J + ks, j)$ with probability $u_s^{i,j}, s = 1, \ldots, Q$.

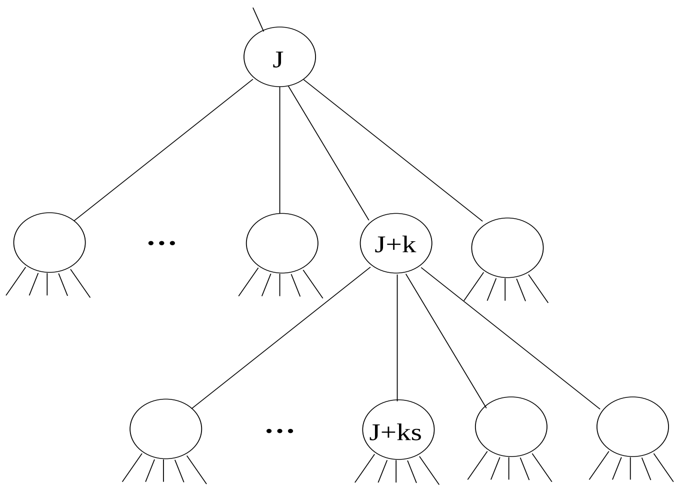


Figure 3.3: *A tree structured Markov chain and its transitions*

Define $m \times m$ matrices $D_k, A_{k,s}$ and U_s with respective $(i,j)^{th}$ elements given by $d_k^{i,j}, a_{k,s}^{i,j}$ and $u_s^{i,j}$. Notice that transitions from state $(J+k,i)$ do not dependent upon J, moreover, transitions to state $(J+ks,j)$ are also independent of k. When the Markov chain is in the root state $(\emptyset,i)$ at time t then the state at time $t+1$ is determined as follows:

1. $(\emptyset,j)$ with probability $f^{i,j}$,

2. (k,j) with probability $u_k^{i,j}, k = 1,\dots,Q$.

Define the $m \times m$ matrix F with corresponding $(i,j)^{th}$ element given by $f^{i,j}$. A fundamental period of a tree structured QBD Markov chain that starts in the state $(J+k,i)$ is defined as the first passage time from the state $(J+k,i)$ to one of the m states (J,j) for $j = 1,\dots,m$. In general, Q and m are assumed to be finite. The theory of tree structured QBD Markov chains can be extended for Q and m infinite. However, in order to solve the Markov chain numerically both Q and m need to be finite. For a more detailed description of the notations and algebra see Yeung, *et al* [78].

3.3 Markovian Model for the Basic Binary CTM Algorithm with Free Access

New packets are generated according to a D-BMAP (see Chapter 2) as follows. Assume that the D-BMAP is in some state i, $1 \leq i \leq l$, at time t. Then, with a probability $(B_n)_{i,j}$, the state at time $t+1$ is j and $n \geq 0$ new packets are generated at the boundary of slot $t-1$ and t. Due to the free access these n new packets are transmitted (for the first time) in time slot t by their corresponding stations.

3.3.1 A First Attempt

The system at time slot t is fully specified by the state of the D-BMAP at the boundary of slot t and $t+1$ and by the current stack level for slot t of each active station. The value of all these current stack levels can be specified by a single string $s_k s_{k-1} \ldots s_1 s_0$, where s_i specifies the number of active stations with a current stack level for slot t equal to i. Therefore, the system is fully characterized by the Markov chain (V_t, W_t), where W_t denotes the state of the D-BMAP at the boundary of slot t and $t+1$ and V_t represents the string that holds the current stack level for slot t of all active stations. It is easy to see that (V_t, W_t) is a tree structured Markov chain. Indeed, the node $s_k s_{k-1} \ldots s_1 s_0$ is the parent of the nodes $s_k s_{k-1} \ldots s_1 s_0 s$ for $s \geq 0$. Each node, including the root node $\emptyset$, contains l states (the l states of the D-BMAP) and has an infinite number of children. The root node, denoted as $\emptyset$, represents the case when there are no active stations.

The chain (V_t, W_t) is not of the Quasi-Birth-Death type. For instance, suppose that the chain is in the state (J, i) with $J = 2\ 5$ at time t. Therefore, 5 active stations have their current stack level for slot t at zero, i.e., transmit in slot t, and 2 active stations have their current stack level for slot t at one. Next, suppose that 3 out of the 5 stations increase their current stack level to one as a result of the coin flip procedure. When a colliding station determines to join either the first or the second group, it is said to flip a coin (if $p = 1/2$ a fair coin). The coin flipping of all colliding stations is referred to as the coin flipping procedure. Then, at time $t+1$, the Markov chain is in the state (K, j) with $K = 2\ 3\ (2+s)$ with probability $(B_s)_{i,j}$ (i.e., s new arrivals occurred on the boundary of slot t and slot $t+1$). This type of transitions (to the grandchildren of the parent node) is not allowed in a tree structured QBD Markov chain. Also, the Markov chain is no longer of the $GI/M/1$ type (see Yeung, *et al* [79]) and there is no simple or explicit solution for its stationary distribution.

3.3.2 The Actual Model

In order to solve the problem indicated in the previous section we make the number of stations with a current stack level for slot t equal to zero a part of the auxiliary variable. Active stations that have a current stack level for slot t larger than zero are referred to as backlogged stations (at time t). Consider the following Markov chain (X_t, N_t). Let X_t be the string holding the current stack level for slot t of all backlogged stations (at time t). For instance, when $X_t = s_k \ldots s_2 s_1$ there are $\sum_{i=1}^{k} s_i$ backlogged stations, of which $s_i \geq 0$ have their current stack level for slot t equal to i. In this example there are no stations with a current stack level for slot t larger than k. The sample space of the random variable X_t is $\Omega_1 = \{\emptyset\} \cup \{J : J = s_k \ldots s_1, s_j \geq 0, 1 \leq j \leq k, k \geq 1\}$. Notice, the string J is allowed to have a number of leading zeros (see Note 1 for more comments on this issue). The random variable X_t has a tree structure. For instance, the children of $s_k \ldots s_1$ are $s_k \ldots s_1 s, s \geq 0$. Thus, each node in the tree has an infinite number of children. N_t holds both the number of active stations with a current stack level for slot t equal to zero and the state of the D-BMAP at the boundary of slot t and $t+1$. The sample space of the random variable N_t is $\Omega_2 = \{(i, j) \mid i \geq 0, 1 \leq j \leq l\}$.

It is easy to see that (X_t, N_t) is a Markov chain. The state space of the Markov chain is $\Omega_1 \times \Omega_2$. In order to solve this Markov chain the nodes of X_t should have a finite number of children and the auxiliary variable N_t should have a finite number of states. Therefore, the Markov chain (X_t, N_t) is approximated by another bivariate Markov chain (X_t^d, N_t^d). (X_t^d, N_t^d) is obtained by setting a maximum d on the number of stations that can have the same current stack level for slot t (including stack level zero). If a situation occurs in which $d + k, k > 0$, stations have the same current stack level for slot t, k stations are assumed to drop their packet. Thus, introducing d can cause stations to drop their packet. Packets are otherwise never dropped by a station. Nevertheless, provided that d is chosen sufficiently large there should hardly be any difference between the performance measures of (X_t^d, N_t^d) and (X_t, N_t) (the difference between the recurrence of both Markov chains is discussed in Section 3.5). We state that d is chosen sufficiently large if the ratio of dropped packets due to the introduction of d is smaller than 10^{-9}, i.e., if less than one in a billion packets is dropped. The introduction of the parameter d is the only approximation required to evaluate the basic binary CTM algorithm with free access. There is no obvious relationship between a sufficiently large value for d and the maximum n for which $B_n \neq 0$ (also such an n does not necessarily exist). For instance, a sufficiently large d for the Bulk arrival process with $v = [4]$, as defined in Section 2.1.3, is found for $d \geq 18$ for $L = 10$, $d \geq 12$ for $L = 80$ and $d \geq 10$ for $L = 800$ (whereas $B_n = 0$ for $n \geq 5$ in each of the three cases).

Let us now consider (X_t^d, N_t^d) in more detail. X_t^d is the string that holds the current stack level for slot t of all backlogged stations. For instance, when $X_t^d = s_k \dots s_2 s_1$ then for s_i backlogged stations the current stack level for slot t equals i. The sample space of the random variable X_t^d is $\Omega_1^d = \{\emptyset\} \cup \{J : J = s_k \dots s_1, 0 \leq s_j \leq d, 1 \leq j \leq k, k \geq 1\}$. X_t^d has a tree structure, e.g., $s_k \dots s_1 s, 0 \leq s \leq d$, are children of $s_k \dots s_1$. Therefore, each node in Ω_1^d has $d+1$ children. As opposed to the general description of the tree structured QBD Markov chain (see Section 3.2) we represent the children of a node by 0 to d instead of 1 to $d+1$. N_t^d represents the number of stations that transmit in slot t (i.e., the current stack level for slot t of these stations is zero) and the state of the D-BMAP at the boundary of slot t and $t+1$. The sample space of N_t^d is $\Omega_2^d = \{(i, j) \mid 0 \leq i \leq d, 1 \leq j \leq l\}$. It is easy to see that (X_t^d, N_t^d) is a Markov chain and the state space of the Markov chain (X_t^d, N_t^d) is $\Omega_1^d \times \Omega_2^d$.

We now prove that transitions made by the Markov chain (X_t^d, N_t^d) are either transitions to a child or a parent node (except from the root node $\emptyset$). Assume that the Markov chain (X_t^d, N_t^d) is in node $J + k$ at time t, i.e., $X_t^d = J + k$. If slot t contains a collision of $c \geq 2$ stations—that is, N_t^d is of the form (c, j) with $c \geq 2, 1 \leq j \leq l$—all backlogged stations increment their current stack level by one. Thus, the integers in the string $J + k$ shift one position to the left and $X_{t+1}^d = J + ks$ with $0 \leq s \leq c$ (s of the c colliding stations set their current stack level for slot $t+1$ at one as a result of the coin flip). N_{t+1}^d is determined by j, c and the probability that a station selects the first group p. Thus, a collision in slot t causes the Markov chain to make a transition to a child node (this is also the case for $X_t^d = \emptyset$). If slot t does not hold a collision, all backlogged stations decrement their current stack level by one, i.e., shift one position to the right. Hence, if slot t does not hold a collision, the chain will be in the parent node J at time slot $t+1$ (for $X_t^d = \emptyset$ the chain remains in the root node). In conclusion, the chain can only make transitions from a node to either its parent node or to one of its children.

In order for the Markov chain (X_t^d, N_t^d) to be a tree structured QBD Markov chain the following two additional conditions have to be satisfied. First, the probability of making a transition from state $(J+k, (i,j))$ to state $(J, (i',j'))$ may not dependent upon J. As noted above, such a transition takes place whenever slot t does not hold a collision. Clearly, j', the new state of the D-BMAP, is solely determined by j, the old state of the D-BMAP, and thus independent of J. While, i', the number of stations that transmit in slot $t+1$, is determined by k, the number of stations that decrease their current stack level from one to zero, and j, the old state of the D-BMAP (because this state j determines the number of new arrivals on the boundary of slot t and slot $t+1$). Second, the probability of making a transition from state $(J+k, (i,j))$ to state $(J+ks, (i',j'))$ may not dependent upon J and k. Such a transition occurs whenever slot t does hold a collision. Again, j', the new state of the D-BMAP, is determined by j, the old state of the D-BMAP. While, s, the number of stations that increase their current stack level to one (as a result of the coin flipping), is determined by i and the probability p. Finally, i', the number of stations that transmit in slot $t+1$, is determined by i, p and j, the old state of the D-BMAP (because this state j determines the number of new arrivals).

In conclusion, the Markov chain (X_t^d, N_t^d) is a tree structured QBD Markov chain. A tree structured QBD Markov chain is fully characterized by the matrices D_k, U_s, $A_{k,s}$ and F (see Section 3.2). The matrices $A_{k,s}$ hold the transition probabilities that the chain (X_t^d, N_t^d) goes from state $(J+k, (i,j))$ to the state $(J+s, (i',j'))$. These transitions are transitions between sibling nodes. Two nodes are referred to as sibling nodes if they have the same parent node. Remember that the chain (X_t^d, N_t^d) can only make transitions to its parent or to its children, therefore, the entries of the matrices $A_{k,s}$ are zero. This fact reduces the memory and time requirements of the algorithm that calculates the steady state probabilities of (X_t^d, N_t^d) when it is ergodic (for details see Section 3.4).

The matrices D_k hold the transition probabilities that the chain (X_t^d, N_t^d) goes from state $(J+k, (i,j))$ to the state $(J, (i',j'))$. This happens when slot t does not hold a collision. Therefore, the state i, the number of stations that transmit in slot t, must be equal to zero or one. Moreover, the state i', the number of stations that transmit in slot $t+1$, equals k, the number of stations that decrease their current stack level from one to zero, plus some possible new arrivals. Hence,

$$D_k((i,j),(i',j')) = \begin{cases} (B_{i'-k})_{j,j'} & i \leq 1, i' \geq k, i' < d, \\ \sum_{n \geq d-k} (B_n)_{j,j'} & i \leq 1, i' \geq k, i' = d, \\ 0 & otherwise, \end{cases} \tag{3.1}$$

where $(B_n)_{j,j'}$ holds the probability that n new arrivals occur and that the input D-BMAP changes its state from j to j'.

The matrices U_s hold the transition probabilities that the chain (X_t^d, N_t^d) goes from state $(J+k, (i,j))$ to the state $(J+ks, (i',j'))$. This happens when slot t holds a collision. Therefore, the state i, the number of stations that transmit in slot t, must be larger than or equal to 2. Moreover, the state i', the number of stations that transmit in slot $t+1$, equals i, the number of stations that transmitted in slot t, minus s, the number of stations that increase their current stack level to one (as a result of the coin flipping), plus some possible new arrivals. Clearly, s can never be larger than i. Hence,

$$U_s((i,j),(i',j')) = \begin{cases} C_s^i p^{i-s} q^s (B_{i'-(i-s)})_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' < d, \\ C_s^i p^{i-s} q^s \sum_{n \geq d-(i-s)} (B_n)_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' = d, \\ 0 & otherwise, \end{cases} \quad (3.2)$$

where $(B_n)_{j,j'}$ holds the probability that n new arrivals occur and that the input D-BMAP changes its state from j to j' and C_s^i denotes the number of different possible combinations of s from i different items.

Assume that the Markov chain is in node $J = \emptyset$ at time t, i.e., $X_t^d = \emptyset$. Then the transitions to the nodes s, $0 \leq s \leq d$, are governed by the matrices U_s, whereas the transitions to the root node $\emptyset$ are as follows. The matrix F holds the transition probabilities that chain (X_t^d, N_t^d) goes from state $(\emptyset, (i,j))$ to the state $(\emptyset, (i',j'))$. This happens whenever slot t does not hold a collision, i.e., $i \leq 1$. The state i', the number of stations that transmit in slot $t+1$, equals the number of new arrivals (occurring on the boundary of slot t and slot $t+1$). Hence,

$$F((i,j),(i',j')) = \begin{cases} (B_{i'})_{j,j'} & i \leq 1, i' < d, \\ \sum_{n \geq d} (B_n)_{j,j'} & i \leq 1, i' = d, \\ 0 & otherwise, \end{cases} \quad (3.3)$$

where $(B_n)_{j,j'}$ holds the probability that n new arrivals occur and that the input D-BMAP changes its state from j to j'.

Note 1: It is possible that a string J has a number of leading zeros. The semantics of such a string J is identical to that of the string J without the leading zeros. For instance, $J = 0\ 0\ 4\ 0\ 5$ has the same meaning as $K = 4\ 0\ 5$. Strings with leading zeros arise from the following situation. When the Markov chain (X_t^d, N_t^d) is in the root state $J = \emptyset$, i.e., $X_t^d = \emptyset$, a transition might occur to state 0. For instance, suppose that $N_t^d = (c, j)$, with $c \geq 2, 1 \leq j \leq l$, and assume that the current stack level for slot $t+1$ is set at zero for each of the c colliding stations (as a result of the coin flip procedure). Then, at time $t+1$, according to Equation (3.2), the Markov chain (X_t^d, N_t^d) is in the node 0. It might seem more appropriate to remain in the root node $J = \emptyset$ in such cases, or equivalently to avoid strings with leading zeros. If we exclude this type of transitions; that is, eliminate such strings, the node variable X_t^d would have a tree structure where every node has $d+1$ children except for the root node $\emptyset$ (who has d children). In Yeung, *et al* [79] this type of Markov chain is called a Markov chain with a forest structure and algorithms to calculate the steady state are provided. Both approaches lead to the same steady state probabilities (after rearranging the states appropriately). The advantage of allowing this type of transitions is that we get a slightly faster algorithm because the boundary condition is slightly less complicated.

3.4 The Stationary Distribution of the Queue String

According to Yeung and Alfa [78], a matrix geometric solution exists for an ergodic QBD Markov chain with a tree structure. The Markov chain (X_t^d, N_t^d) is aperiodic whenever the D-BMAP modeling the input traffic is aperiodic. The irreducibility is not always inherited from the input D-BMAP, e.g., D-BMAPs with $B_0 = 0$ or $B_n = 0, n \geq 2$. In Section 3.5 we address the problem of determining whether the Markov chain (X_t^d, N_t^d) is positive recurrent. Define, for each string $J \in \Omega_1^d$, $0 \leq i \leq d$ and $1 \leq j \leq l$

$$\pi(J,(i,j)) = \lim_{t \to \infty} P[(X_t^d, N_t^d) = (J,(i,j))]. \tag{3.4}$$

Denote by $\pi(J,i) = (\pi(J,(i,1)),\ldots,\pi(J,(i,l)))$ and by $\pi(J) = (\pi(J,0),\ldots,\pi(J,d))$. In order to calculate the $1 \times l(d+1)$ vectors $\pi(J)$ the following three sets of $l(d+1) \times l(d+1)$ matrices play an important role [78].

Let $G_k, 0 \leq k \leq d$, denote the matrix whose $(i,v)^{th}$ element is the probability that the Markov chain (X_t^d, N_t^d) is in state (J,v) at the end of the fundamental period given that this period starts from state $(J+k,i)$. These matrices are stochastic for recurrent QBD Markov chains with a tree structure (Takine, *et al* [62]). Let $R_k, 0 \leq k \leq d$, denote the matrix whose $(i,v)^{th}$ element is the expected number of visits to $(J+k,v)$ given that $(X_0^d, N_0^d) = (J,i)$ before visiting node J again. Let $V_k, 0 \leq k \leq d$, denote the matrix whose $(i,v)^{th}$ element is the taboo probability that starting from $(J+k,i)$, the chain eventually returns to a node with the same length as $J+k$ by visiting $(J+k,v)$, under the taboo of the node J and the sibling nodes of $J+k$, i.e., the nodes $J+s, s \neq k$.

Yeung and Alfa [78] have shown that the matrices G_k and R_k can be expressed in terms of V_k. Moreover, if a tree structured QBD Markov chain does not allow transitions between sibling nodes, they were able to shown that the following simple expressions hold:

$$\begin{aligned} G_k &= (I - V_k)^{-1} D_k, && (3.5)\\ R_k &= U_k (I - V_k)^{-1}, && (3.6)\\ V_k &= A_{k,k} + \sum_{s=0}^{d} U_s G_s. && (3.7) \end{aligned}$$

If however transitions between sibling nodes were allowed it would still be possible to solve the chain but the equations would be more complicated and the resulting iterative scheme more time consuming [78, 79]. Notice that the matrices $V_k, 0 \leq k \leq d$, are identical if the matrices $A_{k,k}, 0 \leq k \leq d$, are identical. For the Markov chain (X_t^d, N_t^d) the matrices $A_{k,k}, 0 \leq k \leq d$, are equal to zero, therefore the matrices $V_k, 0 \leq k \leq d$, are identical. In the remaining part of this section we drop the subscript k if we refer to V_k. Using equations (3.5) and (3.7), we obtain

$$V = \sum_{s=0}^{d} U_s (I - V)^{-1} D_s. \tag{3.8}$$

As a special case of Theorem 2 in Yeung and Alfa [78], the matrix V can be obtained as $\lim_{N\to\infty} V[N]$ from the recursion

$$V[N+1] = \sum_{s=0}^{d} U_s(I - V[N])^{-1} D_s, \tag{3.9}$$

where $V[0] = 0$. Also, the matrices $G_s[N] = (I - V[N])^{-1} D_s$ converge to the substochastic matrices G_s. Since we do not know in advance whether the Markov chain (X_t^d, N_t^d) is recurrent, we do not use the possible stochastic nature of the matrices G_s as a stopping criterion for the recursion in (3.9). We simply repeat the recursion until all matrices $G_s[N], 0 \leq s \leq d$, have stabilized.

Next, the matrices $R_k, 0 \leq k \leq d$, are calculated from the matrix V using equation (3.6). The steady state vectors $\pi(J)$ are then calculated as follows [78]:

$$\pi(J+k) = \pi(J) R_k, \tag{3.10}$$

where $\pi(\emptyset)$ is the left invariant vector of the matrix $F+V$, i.e., $\pi(\emptyset)(F+V) = \pi(\emptyset)$, and $\pi(\emptyset)$ is normalized as $\pi(\emptyset)(I-R)^{-1}\mathbf{e} = 1$. The matrix R is defined as $\sum_{s=0}^{d} R_s$. In order to clarify the subsequent steps required to calculate the steady state probabilities we have summarized them in the following algorithm:

Algorithm:

- INPUT: A sequence of matrices $B_n, n \geq 0$, that characterize the D-BMAP input traffic.
- STEP 1: Calculate the matrices $D_k, 0 \leq k \leq d$, $U_s, 0 \leq s \leq d$, and F by making use of formulas (3.1), (3.2) and (3.3).
- STEP 2: Determine the matrix V using the iterative formula presented in (3.9).
- STEP 3: Calculate the matrices $R_k, 0 \leq k \leq d$, by means of equation (3.6).
- STEP 4: Determine the vector $\pi(\emptyset)$ as follows: $\pi(\emptyset) = \pi(\emptyset)(F+V)$, where $\pi(\emptyset)$ is normalized as $\pi(\emptyset)(I-R)^{-1}\mathbf{e} = 1$.
- STEP 5: Calculate de steady state probabilities of interest using the equation $\pi(J+k) = \pi(J)R_k$.

REMARK: At the end of STEP 4 one can determine whether the parameter d was chosen sufficiently large (see Note 2), if not, d has to be increased and the first four steps have to be repeated, i.e., everything has to be recalculated. For many numerical examples setting d as small as 10 was sufficient (see Section 3.7). Thus, one starts with $d = 2$ and repeats the first 4 steps until d is sufficiently large. It is however possible to reduce the the computational effort by making a first estimate for the starting value of d (instead of $d = 2$). If we estimate the value of d larger than the smallest possible d for which d is sufficiently large, we are finished after one run. One must however note that the larger we choose d, the more time it requires to compute the first four steps. Therefore, one should try to limit the margin of overestimation. During the numerical trials we noticed that there exists a strong relationship between a sufficiently large d and the burstiness, i.e., the variation of the number of arrivals in a time slot, of the input process. We used the following heuristic method to reduce the computation times: if $d = x$ was sufficiently large for a specific D-BMAP and the next D-BMAP we are about to evaluate is more, resp. less, bursty we make use of a larger, resp. smaller, first estimate for a sufficiently large d.

Note 2: We can make use of the following test to determine whether d was chosen sufficiently large. Let ρ be the load, i.e., arrival rate λ, of the D-BMAP modeling the aggregated input traffic. From the steady state probabilities we can calculate $\sum_{J,j} \pi(J,(1,j))$. This sum is, due to the law of total probability, equal to the probability that there is exactly one active station with a current stack level for slot t equal to zero. Therefore, this sum matches the probability of having a successful transmission. We can now compare this with the arrival rate λ, i.e., load ρ, of the D-BMAP to get a value for the ratio of dropped packets. In conclusion, we state that d is chosen sufficiently large whenever $(\rho - \sum_{J,j} \pi(J,(1,j)))/\rho < 10^{-9}$.

3.5 Stability Issues

In Chapter 1 we mentioned that Mathys and Flajolet [43] have shown that the basic binary CTM algorithm with free access is stable under a Poisson flow of arrivals if the arrival rate $\lambda < .360177$ (using fair coins, i.e., for $p = 1/2$). In this section we indicate how to determine whether the basic binary CTM algorithm with free access is stable under D-BMAP traffic. Define $\mathcal{S}$ as the set of all (primitive) D-BMAPs. $\mathcal{S}$ can be split into two subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ such that the CTM algorithm with free access is stable for $s \in \mathcal{S}_1$ and is unstable for $s \in \mathcal{S}_2$. For instance, the CTM algorithm is stable for all D-MAPs, i.e., D-BMAPs with $B_n = 0$ for $n \geq 2$.

A D-BMAP s belongs to $\mathcal{S}_1$ if and only if the Markov chain (X_t, N_t) is stable, i.e., positive recurrent. To test whether the Markov chain (X_t, N_t) is positive recurrent, we study the stability of the Markov chain (X_t^d, N_t^d). Clearly, the chain (X_t, N_t) is transient whenever the chain (X_t^d, N_t^d) is transient. Indeed, (X_t^d, N_t^d) behaves identical to (X_t, N_t) except that it drops a packet from time to time. Clearly, this only improves the expected delay suffered by an arbitrary packet. The stability of the chain (X_t^d, N_t^d) is however not sufficient to prove that the chain (X_t, N_t) is stable. For instance, for every $s \in \mathcal{S}$, (X_t^1, N_t^1) is stable. Even when d is chosen sufficiently large, it is still possible that the dropping of these rare packets (even when we lose less than one in a billion) causes the chain (X_t^d, N_t^d) to become stable while (X_t, N_t) is not. Hence, it is possible that we slightly overestimate the stability point of a particular arrival process. There exists only one case we can use to get an idea of the margin of overestimation: the Poisson result. Numerical results (not included in Section 3.7) have indicated that for $d = 10$ the overestimation is less than

.000003 (the chain was unstable for $\lambda = .36018$ while the exact result by Flajolet states .360177). Further increasing d would result in even smaller overestimation errors.

The Markov chain (X_t^d, N_t^d) is recurrent if and only if the matrices $G_k, 0 \leq k \leq d$, are stochastic (HE [25]). Provided that the Markov chain (X_t^d, N_t^d) is recurrent, we define a heuristic measure d_s for its stability as follows. Let $\pi(i,j), 0 \leq i \leq d$ and $1 \leq j \leq l$, be the probability that the auxiliary variable N_t^d is equal to (i,j). Hence, $\pi(i,j) = \sum_J \pi(J,(i,j)) = \pi(\emptyset)(I-R)^{-1}$ (see Section 3.4). Let $d_s = \sum_j \pi(0,j) + \sum_j \pi(1,j) - \sum_{j,i>1} \pi(i,j)$. d_s can be seen as the difference between the drift towards the root node and the drift away from the root node. Indeed, $\sum_j \pi(0,j)$ is equal to the probability that slot t is empty, i.e., no transmission takes place in slot t, and $\sum_j \pi(1,j)$ is the probability that slot t holds a successful transmission. Therefore, $\sum_j \pi(0,j) + \sum_j \pi(1,j)$ is the probability that the Markov chain makes a transition to a parent node. While, $\sum_{j,i>1} \pi(i,j)$ represents the probability that a collision takes place in slot t—that is, the chain makes a transition to a child node. The difference between these two probabilities is used as a measure for the stability.

3.6 Performance Measures

Although we mainly focus on the stability characteristics of the basic binary CTM algorithm, we can also obtain a number of other interesting performance measures. As far as the numerical results are concerned we restrict ourselves in this chapter to the stability. Numerical results on the mean delay and other performance measures are presented in Chapter 4 in order to compare the performance of the basic Q-ary CTM algorithm with free access for different values of the splitting factor Q.

3.6.1 The Fundamental Period and Mean Delay

Define $\Phi_1(i,j), 0 \leq i \leq d$ and $1 \leq j \leq l$, as the expected length of a fundamental period given that this period starts from state $(J+k,(i,j))$. Notice that these expected values do not depend upon J and k. $\Phi_1(i,j)$ is the expected number of time slots necessary to resolve a collision of i stations provided that the D-BMAP is in state j (at the end of the time slot in which the i stations collide). Let $\Phi_1(i) = (\Phi_1(i,1), \ldots, \Phi_1(i,l))$ and $\Phi_1 = (\Phi_1(0), \ldots, \Phi_1(d))$. Then, the column vector Φ_1^t (x^t denotes the transposed vector of x) obeys the following equation:

$$\Phi_1^t = \mathbf{e} + \sum_{s=0}^{d} U_s[\Phi_1^t + G_s \Phi_1^t]. \tag{3.11}$$

This equation is obtained as follows. The expected length of the fundamental period equals one if the first slot of the period is collision free, i.e., if i equals zero or one (the first $2(d+1)$ rows of U_s are zero, i.e., $\Phi_1(i,j) = 1$ for $i = 0$ or 1). Otherwise, the first slot holds a collision and the expected length of the fundamental period equals one (the first slot) plus the expected time required to resolve the first group plus the expected time required to resolve the second group. In order to calculate the expected time required to resolve the first group we apply the law of total probability on the state of the D-BMAP at the boundary of the second and third slot of the fundamental period (the state at the boundary of the first and second is j), on the number of colliding stations that select the second group and on the number of new arrivals occurring on the slot boundary of the first and second slot of the fundamental period. In matrix form this leads to $\sum_s U_s \Phi_1^t$. For the expected time required to resolve the second group we also apply the law of total probability on the state of the D-BMAP at the end of the slot following the fundamental period initiated by the first group and on the number of new arrivals on the boundary of the last slot of the fundamental period initiated by the first group and the first of the period initiated by the second group. In matrix form this leads to $\sum_s U_s G_s \Phi_1^t$. Equation (3.11) can be solved as a set of linear equations or using an iterative method.

Define $\Upsilon(k,j), 1 \leq k \leq d$ and $1 \leq j \leq l$, as the probability that $N_t^d = (k,j)$ at an arrival instant. Details on how to calculate $\Upsilon(k,j)$ are provided in Section 3.6.3. Thus, the probability that the transmission of a packet is successful at its first attempt is $\sum_j \Upsilon(1,j)$. Let $U(delay)$ be

$$U(delay) = \sum_{i=1}^{d} \sum_{j=1}^{l} \Upsilon(i,j) \Phi_1(i,j). \tag{3.12}$$

Then $U(delay)$ is an upper bound on the mean delay experienced by an arbitrary packet. It is possible to calculate the mean delay $E(delay)$ as follows.

Define $\Phi_2(i,j), 1 \leq i \leq d$ and $1 \leq j \leq l$, as the expected delay suffered by an arbitrary packet provided that the first transmission of the packet coincided with the transmission of $i-1$ other packets and provided that the D-BMAP is in state j after the first transmission. Let $\Phi_2(i) = (\Phi_2(i,1), \ldots, \Phi_2(i,l))$ and $\Phi_2 = (\Phi_2(0), \ldots, \Phi_2(d))$. The column vector Φ_2^t obeys the following equation (this equation is obtained in a similar manner as Equation (3.11)):

$$\Phi_2^t = \mathbf{e} + \sum_{s=0}^{d} \left(M_s U_s \Phi_2^t + N_s U_s [\Phi_1^t + G_s \Phi_2^t] \right), \tag{3.13}$$

where M_s and N_s are the following $(d+1)l \times (d+1)l$ diagonal matrices:

$$M_s = diag(\mathbf{0}^t, a_1(s)\mathbf{e}^t, \ldots, a_d(s)\mathbf{e}^t), \tag{3.14}$$

$$N_s = diag(\mathbf{0}^t, b_1(s)\mathbf{e}^t, \ldots, b_d(s)\mathbf{e}^t), \tag{3.15}$$

with $a_i(s) = 0$ for $i \leq s$, $a_i(s) = (i-s)/i$ for $i > s$, $b_i(s) = 0$ for $i < s$, $b_i(s) = s/i$ for $i \geq s$, $\mathbf{0}^t$ a $1 \times l$ vector with all elements zero and $\mathbf{e}^t$ a $1 \times l$ vector with all elements equal to one. Remark that $a_i(s)$, resp. $b_i(s)$, represents the probability that our arbitrary packet selects the first, resp. second, group after a collision knowing that s of the colliding stations select the second group. Equation (3.13) can be solved as a set of linear equations or using an iterative method. The expected delay experienced by a packet $E(delay)$ is found as

$$E(delay) = \sum_{i=1}^{d}\sum_{j=1}^{l} \Upsilon(i,j)\Phi_2(i,j). \tag{3.16}$$

3.6.2 Other Performance Measures

Define $\Theta(k,i,j), k \geq 0, 0 \leq i \leq d$ and $1 \leq j \leq l$, as the probability that the highest current stack level held by a station equals k and that the auxiliary variable of the Markov chain (X_t^d, N_t^d) equals (i,j). Let $\Theta(k,j) = (\Theta(k,i,1), \ldots, \Theta(k,i,l))$ and $\Theta(k) = (\Theta(k,0), \ldots, \Theta(k,d))$. Recall that it is possible that a string $J \in \Omega_1$ starts with a sequence of zeros (see Note 1 in Section 3.3.2). Therefore, $\Theta(k) = \sum_{J \in L(k)} \pi(J)$ with $L(k) \subset \Omega_1$, where $L(k)$ is the collection of strings J with a length $m, m \geq k$, and with exactly $m-k$ leading zeros. Define R as $\sum_{i=0}^{d} R_i$, then due to Equation (3.10)

$$\Theta(k) = \pi(\emptyset)(I - R_0)^{-1} \qquad k = 0, \tag{3.17}$$
$$\Theta(k) = \Theta(k-1)(R - R_0) = \pi(\emptyset)(I - R_0)^{-1}(R - R_0) \qquad k = 1, \tag{3.18}$$
$$\Theta(k) = \Theta(k-1)R = \pi(\emptyset)(I - R_0)^{-1}(R - R_0)R^{k-1} \qquad k > 1. \tag{3.19}$$

The matrix $(I - R_0)^{-1} = \sum_j R_0^j$ exists because $R = \sum_i R_i$, $R_i \geq 0$ for $0 \leq i \leq d$ and $(I - R)^{-1} = \sum_j R^j$ exists. Define $\Gamma(k,i,j), k \geq 0, 0 \leq i \leq d$ and $1 \leq j \leq l$, as the probability that the number of backlogged stations equals k and that the auxiliary variable of the Markov chain (X_t^d, N_t^d) equals (i,j). Let $\Gamma(k,j) = (\Gamma(k,i,1), \ldots, \Gamma(k,i,l))$ and $\Gamma(k) = (\Gamma(k,0), \ldots, \Gamma(k,d))$. Then, due to Equation (3.10)

$$\Gamma(k) = \pi(\emptyset)(I - R_0)^{-1} \qquad k = 0, \tag{3.20}$$
$$\Gamma(k) = \sum_{i=1}^{\min(k,d)} \Gamma(k-i)R_i(I - R_0)^{-1} \qquad k > 0. \tag{3.21}$$

Next, define $\Lambda(k), k > 0$, as the expected number of backlogged stations with a current stack level equal to k. The probability of having $i, i > 0$, stations with a current stack level equal to $k, k > 0$, is $\sum_{J \in T(k)} \pi(J)\mathbf{e}$, where the subset $T(k) \subset \Omega_1$ is the collection of strings J for which the k-th integer from the right equals i. Hence,

$$\Lambda(k) = \sum_{i=1}^{d} i\pi(\emptyset)(I - R)^{-1}R_iR^{k-1}\mathbf{e}. \tag{3.22}$$

Define $E[r]$ as the expected number of transmissions required to transmit a packet successfully. $E[r]$ is significantly smaller than $E(delay)$ because an active station only transmits whenever its current stack level is equal to zero. Let $\pi(\emptyset)(I - R)^{-1} = (\alpha(0), \ldots, \alpha(d))$, where $\alpha(i), 0 \leq i \leq d$, is a $1 \times l$ vector. Then, $E[r]$ is found as the ratio of the expected number of transmissions in slot t and the expected number of successful transmissions in slot t

$$E[r] = \frac{\sum_{k=1}^{d} k\alpha(k)\mathbf{e}}{\alpha(1)\mathbf{e}}. \tag{3.23}$$

Finally, let p_e, resp. p_s, resp. p_c, be the probability that a time slot is empty, resp. holds a successful transmission, resp. holds a collision. Then,

$$p_e = \alpha(0)\mathbf{e}, \tag{3.24}$$

$$p_s = \alpha(1)\mathbf{e}, \tag{3.25}$$

$$p_c = \sum_{i=2}^{d} \alpha(i)\mathbf{e}. \tag{3.26}$$

3.6.3 The State of the Auxiliary Variable at Arrival Times

Basically, $\Upsilon(k,j), 1 \leq k \leq d$ and $1 \leq j \leq l$, equals the probability that the first transmission of a packet coincides with the transmission of $k-1$ other packets and that the state of the D-BMAP modeling the input traffic is j after this first transmission. Let $\alpha_s = \pi(\emptyset)(I-R)^{-1}R_s, 0 \leq s \leq d$, and $\alpha = \pi(\emptyset)(I-R)^{-1}$. Clearly, α_s and α are $1 \times l(d+1)$ vectors. Thus α_s can be written as $\alpha_s = (\alpha_s(0), \ldots, \alpha_s(d))$, where $\alpha_s(i), 0 \leq i \leq d$, are $1 \times l$ vectors. Similarly, $\alpha = (\alpha(0), \ldots, \alpha(d))$ and $\pi(\emptyset) = (\pi_0(\emptyset), \ldots, \pi_d(\emptyset))$.

Both equations presented below are a natural extension of the common method used in an M/G/1 type of Markov chain to calculate the steady state probabilities of the Markov chain at an arrival instant given the steady state probabilities at an arbitrary time instant (see, e.g., [4]) and by observing that the packets that are dropped (due to d) are dropped before their first transmission attempt (see Equations (3.1), (3.2) and (3.3)). Let $\Upsilon(k) = (\Upsilon(k,1), \ldots, \Upsilon(k,l))^t$. For $k < d$, we get

$$\begin{aligned} \Upsilon(k) = & \frac{1}{p_s}\Bigg[\sum_{i=0}^{1}\left(\pi_i(\emptyset)kB_k + \sum_{s=0}^{k}\alpha_s(i)(k-s)B_{k-s}\right) + \\ & \sum_{i=2}^{d}\alpha(i)\sum_{s=0}^{\min(i,k)} C_s^i p^s(1-p)^{i-s}(k-s)B_{k-s}\Bigg], \end{aligned} \tag{3.27}$$

where p_s was defined in Section 3.6.2. For $k = d$, we have

$$\begin{aligned} \Upsilon(d) = & \frac{1}{p_s}\Bigg[\sum_{i=0}^{1}\left(\pi_i(\emptyset)d\sum_{j\geq d}B_j + \sum_{s=0}^{d}\alpha_s(i)(d-s)\sum_{j\geq d}B_{j-s}\right) + \\ & \sum_{i=2}^{d}\alpha(i)\sum_{s=0}^{\min(i,k)} C_s^i p^s(1-p)^{i-s}(d-s)\sum_{j\geq d}B_{j-s}\Bigg]. \end{aligned} \tag{3.28}$$

3.7 Numerical Examples

To test whether the Markov chain (X_t^d, N_t^d) is stable, we calculate the matrices $G_k, 0 \leq k \leq d$, and check whether they are stochastic. The matrices G_k are determined by an iterative algorithm which is performed in a floating point environment; hence, the resulting matrices are never "truely" stochastic. Therefore, if all the row sums of G_k are between $1-10^{-9}$ and 1, we conclude that G_k is stochastic. If there is a row in G_k for which the row sum is below $1 - 10^{-4}$ we conclude that the matrix G_k is not stochastic. If the smallest row sum of G_k is between $1 - 10^{-4}$ and $1 - 10^{-9}$ we conclude that the stochastic nature of G_k is undetermined (i.e., the recurrence of the chain (X_t^d, N_t^d) is unclear). Notice that if (X_t^d, N_t^d) is transient we can use the value of the smallest row sum d_t as a heuristic measure of instability.

As with many of the iterative formulas used in the matrix analytical approach [18, 34, 46, 55, 75–77], the number of iterations required by formula (3.9) increases significantly when the Markov chain (X_t^d, N_t^d) is close to instability (e.g., 10 to 100 iterations suffice for many stable and unstable Markov chains, while the number of iterations can become as large as a few thousands when the chain is (very) close to the instability point). This limits the precision by which instability points can be determined.

Next, we determine the instability point of a number of D-BMAP arrival processes presented in Section 2.1.3 for $p = 1/2$. The issue of using biased coins ($p \neq 1/2$) is briefly discussed at the end of this section. In the remainder of this chapter, the instability point is also referred to as the stability point as this is the point where the CTM algorithm switches between being stable and unstable. For most of the numerical results presented below the parameter d was sufficiently large for $d \geq 10$ (see Section 3.3.2 and Note 2 in Section 3.4).

3.7.1 The Discrete Time Poisson Process

From Chapter 1 it follows that the basic binary CTM algorithm with free access is stable for $\lambda < .360177147$ under Poisson input traffic. We start by confirming this result using our analytical model. The results are presented in Table 3.1. The first column of Table 3.1 represents the arrival rate of the input D-BMAP λ, the second indicates whether the chain (X_t^d, N_t^d) is stable or not (S = stable, U = unstable) and the last column represents the heuristic stability measure d_s or the instability measure d_t depending on whether the Markov chain was stable or not. According to Table 3.1 the Markov chain (X_t^d, N_t^d) becomes unstable for λ somewhere between .36015 and .3602. This is in complete correspondence with the results obtained by Mathys and Flajolet [43]. Additional runs have shown that the stability point is found in the interval $[.36015, .36018]$.

3.7.2 The Discrete Time Erlang Process

The discrete time Erlang process was introduced in Section 2.1.3. The stability points for the Erlang process with $k = 2, 3$ and 4 have been determined and the results are presented in Table 3.2. The results indicate that increasing the parameter k results in a higher stability point. This is not surprising because the Erlang distribution becomes more deterministic when increasing k. As a function of k, the growth of the stability point decreases as k increases (this seems logical as the variance of the Erlang distribution

λ	S/U	d_s/d_t
.10000	S	.9745
.30000	S	.5207
.35000	S	.1215
.35500	S	.0617
.36000	S	.0023
.36010	S	.0010
.36015	S	.0003
.36020	U	.9991
.36030	U	.9951
.36050	U	.9872
.36100	U	.9678
.36250	U	.9120
.37000	U	.6791
.40000	U	.2169

Table 3.1: *Stability under Poisson traffic.*

$\lambda = \lambda_e/k$	k	S/U	d_s/d_t
.3625	2	S	.1035
.3650	2	S	.0199
.3655	2	S	.0017
.3656	2	U	.9965
.3658	2	U	.9835
.3660	3	S	.1203
.3670	3	S	.0468
.3675	3	S	.0059
.3676	3	U	.9973
.3680	3	U	.9646
.3675	4	S	.1313
.3682	4	S	.0246
.3684	4	U	.9955
.3690	4	U	.9384

Table 3.2: *Stability under Erlang k traffic.*

decreases linearly in k). For instance, the stability point of the Erlang process with $k = 15$ is below .37. Therefore, the difference between the stability point for the Erlang process with $k = 1$ and $k = 15$ is less than .01, while the variance of the interarrival times is 15 times as large for $k = 1$ as opposed to $k = 15$.

3.7.3 The Discrete Time Markov Modulated Poisson Process

The discrete time Markov modulated Poisson process was introduced in Section 2.1.3. For the numerical examples we restrict ourselves to the interrupted Poisson processes (IPPs). An IPP is an MMPP with two states and the arrival rate corresponding to one of the states, say λ_1, is zero. The IPPs are the most bursty of all MMPPs with two states and are therefore expected to produce the most deviating results from the Poisson result. For instance, the algorithm under $M(\cdot, 2\lambda_1, 30, 30)$ input is stable for $\lambda = 3\lambda_1/2 < .359$; unstable $\lambda > .36$. That is, the stability point is found in the interval [.359, .36].

$\lambda = \lambda_2/2$	S/U	d_s/d_t
0.3250	S	0.0673
0.3400	S	0.0222
0.3450	S	0.0072
0.3466	S	0.0025
0.3480	U	0.9965
0.3500	U	0.9843
0.3600	U	0.9279

Table 3.3: *Stability under $M(0, \cdot, 300, 300)$ input traffic.*

$\lambda = \lambda_2/8$	S/U	d_s/d_t
0.3400	S	0.0202
0.3450	S	0.0056
0.3460	S	0.0027
0.3466	S	0.0009
0.3480	U	0.9952
0.3500	U	0.9856
0.3600	U	0.9449

Table 3.4: *Stability under $M(0, \cdot, 210, 30)$ input traffic.*

$\lambda = v_1/(L+1)$	v_1	S/U	d_s/d_t
0.344800	2	S	0.0161
0.347826	2	S	0.0026
0.348432	2	U	0.9987
0.350900	2	U	0.9008
0.342800	3	S	0.0259
0.349040	3	S	0.0024
0.349854	3	U	0.9926
0.352900	3	U	0.8446
0.347800	4	S	0.0033
0.348432	4	S	0.0012
0.349040	4	U	0.9916
0.350900	4	U	0.9287

Table 3.5: *Stability under the Bulk arrival process.*

$\lambda = \sum v_i/(L+2)$	$\sum v_i$	S/U	d_s/d_t
0.348800	2+1	S	0.0046
0.349854	2+1	S	0.0005
0.350050	2+1	U	0.9969
0.350400	2+1	U	0.9803
0.348400	3+1	S	0.0017
0.348735	3+1	S	0.0006
0.349040	3+1	U	0.9953
0.350900	3+1	U	0.9330
0.344800	2+2	S	0.0090
0.346620	2+2	S	0.0026
0.347826	2+2	U	0.9838
0.348400	2+2	U	0.9631

Table 3.6: *Stability under the Bulk arrival process.*

Tables 3.3 and 3.4 show that the interval [.3466, .348] includes the stability point of both the $M(0,\cdot,300,300)$ and $M(0,\cdot,210,30)$. Thus, although the second IPP is by far the more bursty of the two—because the arrivals are concentrated in 12.5% of the time slots compared to the 50%—their stability point differs less than .0014. As for the influence of correlation, we found that the interval [.348, .349] contains the stability point of the IPP with $a = b = 30$, i.e., the $M(0,\cdot,30,30)$ process. Comparing this with the results in Table 3.3, we see that correlation slightly decreases the stability of the basic binary CTM algorithm with free access (in our example less than .0024). This observation was confirmed by other numerical examples.

3.7.4 The Bulk Arrival Process

The Bulk arrival process is defined in Section 2.1.3. Table 3.5 presents the results for $m = 1$ with $v = [2], [3]$ and $[4]$; whereas Table 3.6 holds the results for $m = 2$ with $v = [2,1], [3,1]$ and $[2,2]$. For each of these processes we gradually decrease L, i.e., increase the arrival rate λ, until the basic binary CTM algorithm with free access becomes unstable. Perhaps somewhat surprisingly: the $v = [2,2]$ process is the first of the six processes to become unstable ($\lambda \in [.346620, .347826]$), then the $v = [2]$ process, followed by either the $v = [4]$ or the $v = [3,1]$ process (we did not attempt to distinguish these two processes), next the $v = [3]$ process and finally the $v = [2,1]$ process. From these results it follows that it is not always the most bursty process that results in the lowest stability point.

3.7.5 Summary for Fair Coins

The stability point of the basic binary CTM algorithm with free access under D-BMAP input depends upon the exact definition of the input process. For instance, the discrete time Poisson process, the Erlang processes, the Markov modulated Poisson processes and the Bulk arrival processes all result in a different stability point. Moreover, it is often difficult to state *a priori*—from the characteristics of the D-BMAPs—which of two input processes results in a higher stability point, i.e., maximum stable throughput. Hence, the stability results of the basic binary CTM algorithm with blocked access are much more transparent as opposed to the free access scheme (see Theorems 2.1 and 2.2 in Section 2.3).

On the other hand, the stability point, i.e., maximum stable throughput, of the free access scheme under D-BMAP input is never far below the stability point under Poisson input (in our examples: at most .014). Thus, the basic binary CTM algorithm with free access seems to maintain its good stability characteristics under D-BMAP input traffic. Clearly, we can always define a D-BMAP with a load $0 \leq \rho \leq 1$ for which the CTM algorithm with free access is stable, for example a D-MAP. Also, although correlation in the input traffic reduces the stability point somewhat, it does not devastate the stability.

An interesting open problem related to this is whether there exists an arrival rate λ_{min} such that the basic binary CTM protocol with free access (with $p = 1/2$) is stable under all primitive D-BMAPs with an arrival rate $\lambda < \lambda_{min}$. During the numerical trials, we did not find a D-BMAP with an arrival rate smaller than $\lambda = .34657 = \ln(2)/2$ for which the basic binary CTM algorithm with free access became unstable. For instance, the $v = [2, 2, 2, 2]$, $v = [2, 2, 2, 2, 2]$, $v = [5]$, $v = [10]$ Bulk arrival processes, the IPP with $a = b = 3000$ and many others turned out to be stable for an arrival rate of $\ln(2)/2$. The value $\ln(2)/2$ is no stranger to the basic binary CTM algorithm because in Section 2.3 we have shown that the basic binary CTM algorithm with blocked access under primitive D-BMAP input traffic is stable for $\lambda < \ln(2)/2 - 10^{-5}$; unstable for $\lambda > \ln(2)/2 + 10^{-5}$. Moreover, the expected length of a busy period initiated by a collision of n stations increases asymptotically as $2n/\ln(2)$ provided that no new arrivals occur. This result also indicates that the Bulk arrival process $v = [n]$ with a load smaller than $\ln(2)/2$ is unlikely to cause instability even for large values of n and L. The question raises whether it is at all possible to find a primitive D-BMAP with an arrival rate $\lambda < \ln(2)/2$ that makes the basic binary CTM algorithm with free access unstable. If not, the basic binary CTM algorithm with free access results in a maximum stable throughput that is at least as good as the corresponding scheme with blocked access under primitive D-BMAP traffic. We therefore formulate the following conjecture:

CONJECTURE **3.1** *The basic binary CTM algorithm with free access is is stable under primitive $(B_n)_n$ D-BMAP traffic if*

1. $\lambda < \ln(2)/2$, *with λ the mean arrival rate,*
2. $(B_n)_n$ *has a finite number of states l.*

3.7.6 Using Biased Coins

Fair coins are the optimal coins for the basic binary CTM algorithm combined with both the free and blocked access strategy provided that the input process is Poisson (see

$PP(\cdot)$		$M(0,\cdot,30,30)$		$ER(\cdot,2)$	
p	$\alpha(d_s)$	p	$\alpha(d_s)$	p	$\alpha(d_s)$
.6000	.351	.5500	.343	.6000	.359
.5500	.358	.5000	.348	.5500	.364
.5200	.359	.4800	.349	.5300	.365 (.0211)
.5100	.360 (.0012)	.4700	.350 (.00047)	.5200	.365 (.0264)
.5000	.360 (.0023)	.4650	.350 (.00062)	.5150	.365 (.0270)
.4900	.360 (.0012)	.4600	.350 (.00060)	.5100	.365 (.0261)
.4800	.359	.4500	.350 (.00011)	.5000	.365 (.0199)
.4500	.358	.4400	.349	.4800	.364
.4000	.351	.4200	.348	.4500	.362

Table 3.7: *The influence of using biased coins on the stability of the basic binary CTM algorithm with free access.*

Chapter 1). Moreover, the stability under primitive D-BMAP traffic is identical to the Poisson stability (see Section 2.3) if the blocked access strategy is used. Hence, fair coins are again optimal. In this section, we investigate whether this result is also valid if the free access strategy is used—that is, whether the basic binary CTM algorithm with free access performs best under D-BMAP input traffic if fair coins are used ($p = 1/2$).

For each arrival process a considered, we vary the probabilities p and $q = 1 - p$, and determine the stability point that corresponds to the couple (a, p). Define α as a multiple of .001 such that the interval $]\alpha, \alpha + 0.001[$ that holds the stability point of (a, p). When the stability point of different couples (a, p) lies within the same interval $]\alpha, \alpha + 0.001[$, we also add the stability measure d_s to determine which value for p performs best. A larger value for d_s implies a more stable Markov chain. Table 3.7 represents the stability points, i.e., maximum stable throughput, as a function of p for the basic binary CTM algorithm with free access under Poisson input traffic, Markov modulated Poisson input traffic and Erlang input traffic. The Poisson result obtained by Mathys and Flajolet [43] is confirmed by our analytical model.

Table 3.7 indicates that the optimal value for p for the $ER(\cdot, 2)$ lies somewhere in the interval $].51, .52[$, whereas the optimal value for the $M(0, \cdot, 30, 30)$ input traffic is found in the range $].46, .47[$. We already mentioned that the optimum for Poisson input is $p = .5$. Thus, the more bursty the input traffic the lower the optimal value of p becomes. Intuitively, this can be understood as follows: the more bursty the input traffic becomes the better it is to postpone the retransmission of some of the colliding packets. For instance, if a collision occurs, in slot t, under Erlang traffic it is more likely that no new arrivals will occur in the next slot, slot $t+1$, as opposed to the slots $t+i, i > 1$. Therefore, it is better to choose p slightly larger than .5. Whereas for the Markov modulated Poisson traffic it is more likely that the D-BMAP is transmitting at a higher rate whenever a collision occurs and therefore it might be interesting to postpone some of the arrivals that occur during this high rate period to a period where a lower input rate is being used (i.e., the probability that new arrivals occur in slot $t + 1$ is larger than in slot $t + i, i > 1$). This line of reasoning also corresponds with the Poisson result: if a collision occurs in slot t, the probability of having a new arrival in slot $t+i$ is identical for all $i > 0$ ($= 1 - e^{-\lambda}$). Therefore, there is no reason to prefer the next slot above any of the other slots, i.e., $p = .5$ is the optimum. Another remark is that the stability point of a single state D-BMAP (i.e., $l = 1$) arrival process remains identical if we swap the value for p and q, e.g., the Poisson results in Table 3.7. Indeed, swapping both values changes the order in which the two sets of colliding stations are resolved. The order is unimportant if the number of arrivals occurring in consecutive time slots is independent.

In conclusion, for bursty and correlated arrival patterns higher throughput results can be achieved by decreasing p. It is however hard to predict the optimal value for p because it depends upon the statistical properties of the arrival process.

3.8 Conclusions

In this chapter we demonstrated that the stability of the basic binary CTM algorithm with free access under D-BMAP input traffic can be determined by constructing a Quasi-Birth-Death (QBD) Markov chain with a tree structure. The following conclusions were drawn from the numerical examples. First, the maximum stable throughput achieved by the basic binary CTM algorithm with free access differs from one arrival process to the other. Hence, the stability is not as transparent as its blocked access counterpart (see Theorem 2.1 and 2.2). Second, correlated and bursty arrival processes tend to result into a smaller maximum stable throughput. However, the maximum stable throughput is never far below the Poisson result. Moreover, we did not find a primitive D-BMAP with an arrival rate $\lambda < \ln(2)/2$ for which the basic binary CTM algorithm with free access ($p = 1/2$) is unstable. The question raises whether it is at al possible to find such a primitive D-BMAP. If not, the basic binary CTM algorithm with free access (and fair coins) outperforms its blocked access counterpart (see Section 2.3) under primitive D-BMAP input traffic. We believe that this is the case because we managed to find many different arrival processes with a rate $\lambda, \ln(2)/2 = .34657 < \lambda < .348$, that resulted in an unstable algorithm, but none with $\lambda < \ln(2)/2$. Moreover, increasing the correlation or burstiness of a specific arrival process often resulted in a decrease of the maximum stable throughput that seemed to converge to the value $\ln(2)/2$, e.g., the Markov modulated Poisson processes. Nevertheless, it could be that we have been looking at the wrong set of arrival processes ☺. Finally, fair coins are no longer the optimal coins for the basic binary CTM algorithm with free access under D-BMAP input, as opposed to the Poisson input case or the blocked access scheme. The correlation between the number of arrivals in slot t and $t+1$ is an important indication as to which coins are optimal. For instance, if there is no correlation one expects fair coins to be optimal, e.g., the Poisson process; while the larger, resp. smaller, the correlation is the smaller, resp. larger, the optimal p is expected to be.

Chapter 4

The Basic and Modified Q-ary CTM Algorithm with Free Access

In this chapter we extend the techniques presented in the previous chapter in order to study the stability of the basic and modified Q-ary CTM algorithm with free access. Thus, we indicate how to construct a tree structured QBD Markov chain that is recurrent, resp. transient, whenever the tree algorithm of interest is stable, resp. unstable. We start with a detailed description of the basic and the modified Q-ary CTM algorithm with free access. Next, in Section 4.2, we introduce the tree structured QBD Markov chains of interest. Numerical results are presented in Section 4.3 and conclusions are drawn in Section 4.4. The work presented in this chapter is to appear in [69]

4.1 The Basic and Modified Q-ary CTM Algorithm

In a first subsection we describe the basic Q-ary CTM algorithm with free access, in a second the modified Q-ary CTM algorithm with free access. We start with a summary of the common features of both algorithms. A single channel (bus, cable, broadcast medium) is shared among many users (sources, nodes, stations) that transmit packetized messages. Time is slotted and transmissions can only occur at the beginning of a time slot. Each time slot has a fixed duration equal to the time required to transmit a packet. Each transmission is within the reception range of every user (in a wireless centralized LAN environment the Base Station could broadcast the result of each uplink transmission). The CTM algorithm is a collision resolution algorithm (CRA) for which each user strives to retransmit its colliding packet till it is correctly received. The users have to resolve this contention without the benefit of any additional source of information on other users' activity.

The CTM protocol separates users that collide recursively—according to some randomization procedure—into distinct groups. The users of the first group retransmit in the next slot, while the users of the i-th group, $i > 1$, wait until the first $i - 1$ groups are resolved. The CTM algorithm is conveniently implemented by letting each user maintain a current stack level (that is, an integer value). Users that have a packet ready to transmit are referred to as active users. Each active user maintains a current stack level (an integer value) and at the end of each time slot the current stack level is updated. The value of the current stack level defines when and if a stations is allowed to (re)transmit a packet. The basic and modified Q-ary CTM algorithms with free access use a different procedure to update the current stack level.

4.1.1 The Basic Q-ary CTM Algorithm with Free Access

The current stack level, that is maintained by each active user, is updated as follows:

- An active user transmits in a time slot t whenever its current stack level for slot t is equal to zero. A user that became active during time slot $t-1$ initializes the current stack level for slot t at zero.
- At the end of a time slot t in which no collision occurs, users with a stack level $i, i > 0$, for slot t set their current stack level for slot $t+1$ at $i-1$ (while a possible successful user becomes inactive).
- At the end of a time slot t in which a collision occurs, all users with a current stack level $i, i > 0$, for slot t set their current stack level for slot $t+1$ at $i+Q-1$. Users with a current stack level for slot t equal to zero split into Q distinct groups: a user joins the i-th group with a probability p_{i-1}. Users that join the i-th group set their current stack level for slot $t+1$ equal to $i-1$.

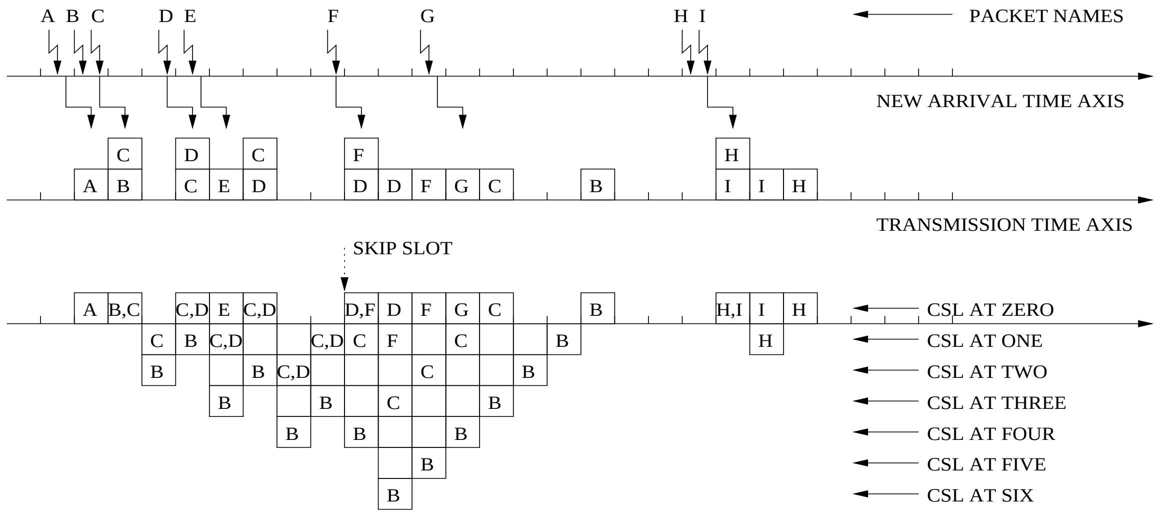
Figure 3.1 shows the state diagram for the basic binary, i.e., $Q = 2$, CTM algorithm with free access; whereas Figure 3.2 presents an example of the transmission process for $Q = 2$. Figure 3.2 also includes a list of group numbers (1 or 2) for each packet to indicate which group the packet joins after each collision (in which it is involved). Thus, the list $1, 2, \ldots$ for packet E indicates that packet E joins the first group as a result of its first collision and the second as a result of its second collision. Selecting one of the Q distinct groups (after a collision) can be seen as flipping a Q-sided coin. A distinction is made between fair coins, i.e., $p_0 = \ldots = p_{Q-1} = 1/Q$, and biased coins. We will consider both fair and biased coins (we do assume that all the stations use the same coins, either fair or biased).

4.1.2 The Modified Q-ary CTM Algorithm with Free Access

The modified CTM algorithm is a well-known improvement of the basic CTM algorithm that skips so-called *doomed* slots (see Chapter 1). *Doomed* slots are slots for which all active stations know *a priori* that the above-mentioned operation of the basic Q-ary CTM algorithm would result in a collision. In order to implement this optimization, ternary feedback (empty, successful or collision slot) is required. As opposed to the basic CTM algorithm where only binary feedback (collision or not) is required. The idea is the following.

Suppose that a collision is followed by $Q-1$ empty slots. This means that all the packets involved in the collision selected the Q-th group. Using the basic CTM algorithm, these stations would transmit in the next slot (together with possible newcomers), generating a guaranteed collision. The modified scheme improves the basic scheme by omitting these slots and by splitting the set of stations that would otherwise result in a guaranteed collision into Q subsets. If the next $Q-1$ slots are again empty, we would get another guaranteed collision and therefore the next slot is again skipped. Thus, whenever, for some $i \geq 1$, the last $1+i(Q-1)$ slots contain a collision followed by $i(Q-1)$ empty slots, this otherwise-wasted slot can be skipped by having all stations immediately act as if it had occurred. This modified scheme is conveniently implemented using a current stack level and a simple count down counter.

Figure 4.1 presents an example of the transmission process for $Q = 3$, it also includes a list of group numbers (1, 2 or 3) for each packet to indicate which group the packet joins after each collision (in which it is involved). Thus, the list $2, 3, 1, 1, \ldots$ for packet D indicates that packet D joins the second group as a result of its first collision, the third as a result of its second collision, the first as a result of its third collision (the skipped collision) and again the first as a result of its fourth collision.



THE SEQUENCE OF GROUPS SELECTED BY THE ASSOCIATED TRANSMITTER:

A: ...	D: 2,3,1,1,...	G: ...
B: 3,...	E: ...	H: 2,...
C: 2,2,3,2,...	F: 2,...	I: 1,...

Figure 4.1: *Example of the Transmission Process: CSL = Current Stack Level*

4.2 Analysis of the Basic and Modified Q-ary CTM Algorithm

This section is subdivided in four parts. Each part describes a tree structured QBD Markov chain that is stable, resp. unstable, whenever either the basic or the modified

CTM algorithm, for specific values of Q, is stable, resp. unstable. The four parts are summarized below:

1. the basic CTM algorithm with $Q > 2$,
2. the modified CTM algorithm with $Q = 2$,
3. the modified CTM algorithm with $Q = 3$,
4. the modified CTM algorithm with $Q > 3$.

With each new part some additional complexity is introduced. In each of these parts new packets are generated according to a D-BMAP characterized by the matrices $(B_n)_n$ (see Chapter 2) as follows. Assume that the D-BMAP is in some state i, $1 \leq i \leq l$, at time t. Then, with a probability $(B_n)_{i,j}$, the state at time $t+1$ is j and n new packets are generated at the boundary of slot $t-1$ and t. Due to the free access these n new packets are transmitted (for the first time) in time slot t by their corresponding stations.

4.2.1 The Basic CTM algorithm with $Q > 2$

As in the previous chapter, we construct a tree structured QBD Markov chain that allows us to study the stability of the basic CTM algorithm with free access, but now for $Q > 2$. In the remainder of this section we indicate how to construct this Markov chain and how to calculate the matrices D_k, U_s, $A_{k,s}$ and F (see Section 3.2) that characterize the Markov chain. These matrices are the input variables of the iterative algorithm described in Section 4.2.5.

Let $q_i, 0 \leq i \leq Q-1$, be the probability that a station increases its current stack level to i, as a result of the coin flipping procedure, provided that it does not increase its current stack level to a value above i. Hence,

$$q_i = \frac{p_i}{1 - \sum_{j>i} p_j}, \tag{4.1}$$

where $p_i, 0 \leq i \leq Q-1$, is the probability that a station increases its current stack level to i as a result of the coin flip.

Consider the stochastic process (X_t, Y_t, Z_t), where X_t denotes the *backlogged string* consisting of the status of all backlogged stations at time slot t, Y_t denotes the number of stations that transmit in time slot t and Z_t denotes the state of the input D-BMAP at the end of time slot t, i.e., at the boundary of slot t and $t+1$. For instance, when $X_t = s_k \ldots s_1$ there are $\sum_i s_i$ backlogged stations, i.e., stations with a current stack level for slot t equal to $i > 0$, and for $s_i \geq 0$ of them the current stack level for slot t equals i. Denote (Y_t, Z_t) as the auxiliary variable N_t. In the previous chapter we have shown that this stochastic process (to be correct its approximation (X_t^d, N_t^d)) is a tree structured QBD Markov chain if $Q = 2$. For $Q > 2$, this process is still a tree structured Markov chain but it is not of the QBD type. For instance, after each slot in which a collision occurs, $Q-1$ integers are added to the backlogged string. These $Q-1$ integers represent the number of stations that increase their current stack level to $1, 2, \ldots, Q-1$ as a result of their coin flipping procedure.

Therefore, we construct an expanded Markov chain $(\mathcal{X}_t, \mathcal{Y}_t, \mathcal{Z}_t, \mathcal{Q}_t)$ and denote $(\mathcal{Y}_t, \mathcal{Z}_t, \mathcal{Q}_t)$ as the auxiliary variable $\mathcal{N}_t$. This expanded Markov chain is constructed such that it is a tree structured QBD Markov chain. The technique used to construct this expanded Markov chain is similar to the method used by Ramaswami [56] in order to reduce an M/G/1-type Markov chain to a QBD Markov chain. The idea behind this expanded Markov chain is that whenever a transition occurs that adds $Q-1$ integers to the node variable X_k, we split this transition into $Q-1$ transitions that each add one integer to the node variable X_k.

Assume a given realization $(X_k(w), N_k(w))$ of the Markov chain (X_k, N_k). The expanded chain $(\mathcal{X}_t, \mathcal{N}_t)$ with $\mathcal{N}_t = (\mathcal{Y}_t, \mathcal{Z}_t, \mathcal{Q}_t)$ is constructed as follows (the range of $\mathcal{Q}_t$ is 0 to $Q-2$). The random variable $\mathcal{Q}_t$ keeps track of how many integers remain to be added to the node variable $\mathcal{X}_t$.

Initial state: If $(X_0(w), N_0(w)) = (J, (i, j))$, then set $(\mathcal{X}_0(w), \mathcal{N}_0(w)) = (J, (i, j, 0))$. Also, set $k = 0$ and $t = 0$, k represents the steps of the original chain and t represents the steps of the expanded chain. We will establish a one-to-one correspondence between the state $(J, (i, j))$ of the original chain and the state $(J, (i, j, 0))$ of the expanded chain.

Transition Rules: We consider three possibilities: $\mathcal{Q}_t(w) = 0$, $\mathcal{Q}_t(w) > 1$ and $\mathcal{Q}_t(w) = 1$.

For $\mathcal{Q}_t(w) = 0$, consider $(X_k(w), (Y_k(w), Z_k(w)))$, and do one of the following:
Case 1: This case corresponds to the situation where the k-th time slot does not hold a collision, i.e., $Y_k(w) \leq 1$. We set $\mathcal{X}_{t+1}(w) = X_{k+1}(w)$ and $\mathcal{N}_{t+1}(w) = (Y_{k+1}(w), Z_{k+1}(w), 0)$. Thus, transitions that do not correspond to a collision remain identical. Next, both t and k are increased by one.
Case 2: This case corresponds to the situation where the k-th time slot does hold a collision, i.e., $Y_k(w) > 1$. Therefore, $X_{k+1}(w)$ can be written as $X_k(w) + s_{Q-1}s_{Q-2}\ldots s_2 s_1$. Then, $(\mathcal{X}_{t+1}(w), \mathcal{N}_{t+1}(w)) = (X_k(w) + s_{Q-1}, (Y_k(w) - s_{Q-1}, Z_k(w), Q-2))$. Indeed, $\mathcal{Q}_{t+1}(w) = Q-2$ because the $Q-2$ integers $s_{Q-2}\ldots s_1$ remain to be added to $\mathcal{X}_{t+1}$. Next, increment both k and t by one.

For $\mathcal{Q}_t(w) > 1$, $X_k(w)$ can be written as $J + s_{Q-1}s_{Q-2}\ldots s_2 s_1$, set $\mathcal{X}_{t+1}(w) = \mathcal{X}_t(w) + s_{\mathcal{Q}_t(w)}$ and $\mathcal{N}_{t+1}(w) = (\mathcal{Y}_t(w) - s_{\mathcal{Q}_t(w)}, \mathcal{Z}_t(w), \mathcal{Q}_t(w) - 1)$. Next, increase t by one and do not alter the value of k.

For $\mathcal{Q}_t(w) = 1$, $X_k(w)$ can be written as $J + s_{Q-1}s_{Q-2}\ldots s_2 s_1$, set $\mathcal{X}_{t+1}(w) = \mathcal{X}_t(w) + s_1$ and $\mathcal{N}_{t+1}(w) = (Y_k(w), Z_k(w), 0)$. Again, increase t by one and do not alter the value of k.

The expanded Markov chain $(\mathcal{X}_t, \mathcal{N}_t)$ is a tree structured QBD Markov chain. The only problem is that every node in $(\mathcal{X}_t, \mathcal{N}_t)$ has an infinite number of children and the auxiliary variable $\mathcal{N}_t$ has an infinite number of states. As in the previous chapter, we can resolve this problem by approximating the expanded chain by the chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ with $\mathcal{N}_t^d = (\mathcal{Y}_t^d, \mathcal{Z}_t, \mathcal{Q}_t))$ that is obtained by putting a maximum d on the number of stations that are allowed to have an identical current stack level.

The expanded Markov chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ does not allow transitions between sibling nodes. Therefore, the entries of the matrices $A_{k,s}$ are zero. Looking at the transition rules described above, the transition blocks D_k and U_s of the Markov chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ are the following.

The matrices D_k hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ goes from state $(J+k, (i,j,m))$ to the state $(J, (i',j',m'))$. This can only happen if $m = 0$, $m' = 0$ and $i \leq 1$. Hence,

$$D_k((i,j,m),(i',j',m')) = \begin{cases} (B_{i'-k})_{j,j'} & m=0, m'=0, i \leq 1, i' \geq k, i' < d, \\ \sum_{n \geq d-k} (B_n)_{j,j'} & m=0, m'=0, i \leq 1, i' \geq k, i' = d, \\ 0 & \textit{otherwise}, \end{cases} \quad (4.2)$$

where $(B_n)_{j,j'}$ holds the probability that n new arrivals occur and that the input D-BMAP changes its state from j to j' (see Chapter 2). Notice that Equation (4.2) is identical to Equation (3.1).

The matrices U_s hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ goes from state $(J+k, (i,j,m))$ to the state $(J+ks, (i',j',m'))$. We separate three different cases. First, assume that $m = 0$. Hence,

$$U_s((i,j,0),(i',j',m')) = \begin{cases} C_s^i q_{Q-1}^s (1-q_{Q-1})^{i-s} (I_l)_{j,j'} & m' = Q-2, i > 1, i' = i-s, \\ 0 & \textit{otherwise}, \end{cases} \quad (4.3)$$

where I_l is an $l \times l$ unity matrix. We simply add the integer, that denotes the number of colliding stations that increase their current stack level to $Q-1$, to the *backlogged string.*

Second, for $m = 1$, we get

$$U_s((i,j,1),(i',j',m')) = \begin{cases} C_s^i q_1^s (1-q_1)^{i-s} (B_{i'-(i-s)})_{j,j'} & m' = 0, \\ & i \geq s, d > i' \geq i-s, \\ C_s^i q_1^s (1-q_1)^{i-s} \sum_{n \geq d-(i-s)} (B_n)_{j,j'} & m' = 0, i \geq s, i' = d, \\ 0 & \textit{otherwise}. \end{cases} \quad (4.4)$$

We add the integer, that denotes the number of colliding stations that increase their current stack level to 1, to the *backlogged string* and allow for new arrivals to join the scheme.

Finally, for $Q-1 > m > 1$, we have

$$U_s((i,j,m),(i',j',m')) = \begin{cases} C_s^i q_m^s (1-q_m)^{i-s} (I_l)_{j,j'} & m' = m-1, i' = i-s, \\ 0 & \textit{otherwise}. \end{cases} \quad (4.5)$$

We add the integer, that denotes the number of colliding stations that increase their current stack level to m, to the *backlogged string.*

4.2.2 The Modified CTM Algorithm with $Q = 2$

Consider the stochastic process (X_t, Y_t, Z_t), where X_t denotes the *backlogged string* consisting of the status of all backlogged stations at time slot t, Y_t denotes the number of stations that transmit in time slot t and Z_t denotes the state of the input D-BMAP at the end of time slot t, i.e., the boundary of slot t and $t+1$. Let $N_t = (Y_t, Z_t)$. For the modified binary CTM algorithm with free access, the stochastic process (X_t, N_t) is not Markovian. We illustrate this by means of an example. Let $X_t = J+k, k > 1$ and $Y_t = 0$. This implies that the t-th time slot is empty and that k stations have a current stack level for slot t equal to one. Consider the following two possibilities for X_{t-1}.

First, let $X_{t-1} = J$ and $Y_{t-1} = k$, in this case slot $t-1$ holds a collision of exactly k stations. A state with $X_t = J+k$ and $Y_t = 0$ is reached if each of the k colliding stations increments its current stack level to one as a result of the coin flip (and no new arrivals occur). After seeing that slot t is empty, all stations know that slot $t+1$ would result in a collision if the basic scheme is used, i.e., slot $t+1$ is a *doomed* slot. As a result, all stations immediately act as if the collision did occur. Therefore, it is possible that $X_{t+1} = J + s$ (if s of the k stations decide to set their current stack level for slot $t+1$ to one as a result of the coin flip).

Second, let $X_{t-1} = J + k + 0$ and $Y_{t-1} = 1$, in which case slot $t-1$ holds a successful transmission. A state with $X_t = J+k$ and $Y_t = 0$ is reached if no new arrivals occur. Due to the success in slot $t-1$, the stations do not consider slot $t+1$ as a *doomed* slot, and the collision in slot $t+1$ will take place. This implies that X_{t+1} is equal to J. In conclusion, the state of the stochastic process (X_t, N_t) at time $t+1$ is not solely determined by the state a time t, which implies that (X_t, N_t) with $N_t = (Y_t, Z_t)$ is not Markovian.

Nevertheless, from the stochastic process (X_t, N_t), we can construct a tree structured QBD Markov chain by adding a value, say -1, to the range of Y_t. $Y_t = -1$ then implies that slot t is empty and that slot $t+1$ would have been a *doomed* slot (if we were using the basic scheme). While $Y_t = 0$ implies that slot t is empty and slot $t+1$ is not considered to be a *doomed* slot. Denote the stochastic process that is obtain by adding -1 to the range of Y_t as (X_t, M_t) with $M_t = (Y_t, Z_t)$. The transitions to and from a state with $Y_t = -1$ are as follows. We enter in a state with $Y_t = -1$ whenever a transition occurs from a collision slot to an empty slot. We stay in a state with $Y_t = -1$ as long as the subsequent slots are empty; otherwise we enter a state with $Y_t \neq -1$.

The stochastic process (X_t, M_t) can be shown to be a tree structured QBD Markov chain (with similar arguments as in Section 3.3.2). However (X_t, M_t) does allow transitions between sibling nodes. This happens whenever an otherwise *doomed* slot is skipped. It is possible to use a more complex (and time consuming) iterative formula (compared to the one in Section 4.2.5), that determines whether a tree structured Markov chain, that does allow transitions between sibling nodes, is stable. Instead, we construct a new tree structured QBD Markov chain $(\mathcal{X}_t, \mathcal{M}_t)$ with $\mathcal{M}_t = (\mathcal{Y}_t, \mathcal{Z}_t)$ that only uses transitions to parent and child nodes. The range of the random variable $\mathcal{Y}_t$ equals $\{(0,n) \mid -1 \leq n\} \cup \{(1,n) \mid 2 \leq n\}$. We will establish a one-to-one correspondence between the states $(J,(i,j))$ of the Markov chain (X_t, M_t) and the states $(J,((0,i),j))$ of $(\mathcal{X}_t, \mathcal{M}_t)$. The idea behind this expanded chain $(\mathcal{X}_t, \mathcal{M}_t)$ is that a transition from a node $J + k$ to a node $J + s$ is split into two transitions: a first one from node $J + k$ to J, followed by a second one from node J to $J + s$. When the transition from node $J + k$ to J takes place we store the value of k in $\mathcal{Y}_t$ by setting $\mathcal{Y}_t = (1, k)$. The fact that the first component of $\mathcal{Y}_t$ is equal to one indicates that the next transition has to be the second step of a split transition.

Assume a given realization $(X_k(w), M_k(w))$ of the Markov chain (X_k, M_k). The expanded chain $(\mathcal{X}_t, \mathcal{M}_t)$ is constructed as follows.

Initial state: If $(X_0(w), M_0(w)) = (J, (i, j))$, then set $(\mathcal{X}_0(w), \mathcal{M}_0(w)) = (J, ((0, i), j))$. Also, set $k = 0$ and $t = 0$, k represents the steps of the original chain and t represents the steps of the expanded chain.

Transition Rules: We consider two possibilities: $\mathcal{Y}_t(w) = (0, i)$ and $\mathcal{Y}_t(w) = (1, i)$.

For $\mathcal{Y}_t(w) = (0, i)$, consider $(X_k(w), M_k(w))$ with $M_k(w) = (Y_k(w), Z_k(w))$, and do one of the following:
Case 1: This case corresponds to the situation where the k-th time slot holds a collision. We set $\mathcal{X}_{t+1}(w) = X_{k+1}(w)$ and $\mathcal{M}_{t+1}(w) = ((0, Y_{k+1}(w)), Z_{k+1}(w))$. Thus, transitions that correspond with a collision remain identical. Next, both t and k are increased by one.
Case 2: This case corresponds to the situation where the k-th time slot does not hold a collision. This implies that $Y_k(w) = 0, 1$ or -1. First, consider $Y_k(w) = -1$. Then $X_k(w)$ can be written as $X_k(w) = J + s$ with $s > 1$ and we get $(\mathcal{X}_{t+1}(w), \mathcal{M}_{t+1}(w)) = (J, ((1, s), Z_k(w)))$. Second, for $Y_k(w) \neq -1$, we get $(\mathcal{X}_{t+1}(w), \mathcal{M}_{t+1}(w)) = (X_{k+1}(w), ((0, Y_{k+1}(w)), Z_{k+1}(w)))$. Hence, the transitions remain identical if $Y_k(w) \neq -1$. Next, increment both k and t by one.

For $\mathcal{Y}_t(w) = (1, i)$, $X_k(w)$ can be written as $J+u$, set $\mathcal{X}_{t+1}(w) = \mathcal{X}_t(w)+u$ and $\mathcal{M}_{t+1}(w) = ((0, Y_k(w)), Z_k(w))$. Next, increase t by one and do not alter the value of k.

As in the previous subsection, we make the number of children in each node and the number of states of the auxiliary variable $\mathcal{M}_t$ finite by putting a maximum d on the number of stations that are allowed to have the same current stack level. Looking at the transitions rules, the transition blocks D_k, $0 \leq k \leq d$, and U_s, $0 \leq s \leq d$, are the following.

The matrices D_k hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ goes from state $(J + k, ((m, i), j))$ to the state $(J, ((m', i'), j'))$. For $m = 0$ and $i \neq -1$, we get

$$D_k(((0,i),j),((m',i'),j')) = \begin{cases} (B_{i'-k})_{j,j'} & m' = 0, i \leq 1, i' \geq k, i' < d, \\ \sum_{n \geq d-k} (B_n)_{j,j'} & m' = 0, i \leq 1, i' \geq k, i' = d, \\ 0 & otherwise. \end{cases} \tag{4.6}$$

Notice, Equation (4.6) is identical to Equation (4.2). For $m = 0$ and $i = -1$, we set

$$D_k(((0,-1),j),((m',i'),j')) = \begin{cases} (B_{i'-k})_{j,j'} & k = 0 \text{ or } 1, m' = 0, i' \geq k, i' < d, \\ \sum_{n \geq d-k} (B_n)_{j,j'} & k = 0 \text{ or } 1, m' = 0, i' \geq k, i' = d, \\ (I_l)_{j,j'} & k > 1, m' = 1, i' = k, \\ 0 & otherwise, \end{cases} \tag{4.7}$$

where I_l is an $l \times l$ identity matrix. A visit to one of the states $(J+k, ((0,-1), j))$, with $k = 0$ or 1, can never occur (the states are transient with an expected return probability equal to 0). Nevertheless, we can still make use of the iterative scheme in Section 4.2.5 by making sure that the probability of eventually returning to a state of the form $(J, ((m,i), j))$ equals one. We realize this by making sure that the corresponding rows of the matrices D_0 and D_1 are stochastic. This explains the somewhat unexpected first two lines in the equation above (we act as if $i = 0$, but any stochastic row will do). For $m = 1$, all entries of $D_k, 0 \leq k \leq d$, are zero.

The matrices U_s hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ goes from state $(J+k, ((m,i), j))$ to the state $(J+ks, ((m',i'), j'))$. For $i \neq s$, we get

$$U_s(((m,i),j),((m',i'),j')) = \begin{cases} C_s^i p_1^s p_0^{i-s} (B_{i'-(i-s)})_{j,j'} & m' = 0, i > 1, \\ & i > s, d > i' \geq i - s, \\ C_s^i p_1^s p_0^{i-s} \sum_{n \geq d-(i-s)} (B_n)_{j,j'} & m' = 0, i > 1, i > s, i' = d, \\ 0 & otherwise. \end{cases} \tag{4.8}$$

For $i = s$, we get

$$U_s(((m,i),j),((m',i'),j')) = \begin{cases} p_1^s (B_0)_{j,j'} & m' = 0, i > 1, i' = -1, \\ p_1^s (B_{i'})_{j,j'} & m' = 0, i > 1, 0 < i' < d, \\ p_1^s \sum_{n \geq d} (B_n)_{j,j'} & m' = 0, i > 1, i' = d, \\ 0 & otherwise. \end{cases} \tag{4.9}$$

Notice that Equation (4.8) and (4.9) are also valid for $m = 0, 1$ and for $i = -1$.

4.2.3 The Modified CTM Algorithm with $Q = 3$

For the basic CTM algorithm with free access we made use of two different models, one for $Q = 2$ and another for $Q > 2$. For the modified CTM algorithm with free access we make use of three different models. Each model description is only valid for the specified range of Q. Rather than going through the entire process that is used to construct the remaining two models, i.e., tree structured QBD Markov chains, we restrict ourselves to a description of the state space of the Markov chains and their corresponding transition probabilities. The techniques used to construct both models are a combination of the methods used to construct the previous two models.

The Markov chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ with $\mathcal{M}_t^d = (\mathcal{Y}_t^d, \mathcal{Z}_t)$, used to study the modified ternary CTM algorithm, is defined on the state space $\Omega_1^d \times (\Omega_2^d \times \Omega_3)$, where $\Omega_1^d = \{\emptyset\} \cup \{J \mid J = s_k \ldots s_1, 0 \leq s_j \leq d, 1 \leq j \leq k, k \geq 1\}$, $\Omega_2^d = \{(0,i) \mid -1 \leq i \leq d\} \cup \{(1,i) \mid 0 \leq i \leq d\} \cup \{(2,i) \mid 2 \leq i \leq d\}$ and $\Omega_3 = \{j \mid 1 \leq j \leq l\}$. The transition matrices D_k, U_s and $A_{k,s}$ are the following. The entries of the matrices $A_{k,s}$ are all zero. Thus, the chain does not allow transitions between sibling nodes.

The matrices D_k hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ goes from state $(J+k, ((m,i),j))$ to the state $(J, ((m',i'),j'))$. For $m=0$ and $i \neq -1$, we get

$$D_k(((0,i),j),((m',i'),j')) = \begin{cases} (B_{i'-k})_{j,j'} & m'=0, i\leq 1, i'\geq k, i'<d, \\ \sum_{l\geq d-k}(B_l)_{j,j'} & m'=0, i\leq 1, i'\geq k, i'=d, \\ 0 & otherwise, \end{cases} \tag{4.10}$$

For $m=0$ and $i=-1$, we set

$$D_k(((0,-1),j),((m',i'),j')) = \begin{cases} (B_0)_{j,j'} & k=0 \text{ or } 1, m'=0, i'=-1, \\ (B_{i'})_{j,j'} & k=0 \text{ or } 1, m'=0, d>i'>0, \\ \sum_{l\geq d}(B_l)_{j,j'} & k=0 \text{ or } 1, m'=0, i'=d, \\ (I_l)_{j,j'} & k>1, m'=2, i'=k, \\ 0 & otherwise, \end{cases} \tag{4.11}$$

where I_l is a $l\times l$ identity matrix. For $m=1$ and 2, all entries of $D_k, 0\leq k\leq d$, are zero.

The matrices U_s hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ goes from state $(J+k, ((m,i),j))$ to the state $(J+ks, ((m',i'),j'))$. For $m=0$ or 2, we get

$$U_s(((m,i),j),((m',i'),j')) = \begin{cases} C_s^i q_2^s(1-q_2)^{i-s}(I_l)_{j,j'} & m'=1, i>1, i\geq s, i'=i-s, \\ 0 & otherwise. \end{cases} \tag{4.12}$$

For $m=1$ and $i>0$, we get

$$U_s(((1,i),j),((m',i'),j')) = \begin{cases} C_s^i q_1^s(1-q_1)^{i-s}(B_{i'-(i-s)})_{j,j'} & m'=0, i\geq s, \\ & d>i'\geq i-s, \\ C_s^i q_1^s(1-q_1)^{i-s}\sum_{l\geq d-(i-s)}(B_l)_{j,j'} & m'=0, i\geq s, i'=d, \\ 0 & otherwise. \end{cases} \tag{4.13}$$

While for $m=1$ and $i=0$, we have

$$U_s(((1,0),j),((m',i'),j')) = \begin{cases} (B_0)_{j,j'} & m'=i=s=0, i'=-1, \\ (B_{i'})_{j,j'} & m'=i=s=0, 0<i'<d, \\ \sum_{l\geq d}(B_l)_{j,j'} & m'=i=s=0, i'=d, \\ 0 & otherwise. \end{cases} \tag{4.14}$$

4.2.4 The Modified CTM Algorithm with $Q>3$

The Markov chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ with $\mathcal{M}_t^d = (\mathcal{Y}_t^d, \mathcal{Z}_t)$, used to study the modified CTM algorithm with $Q>3$, is defined on the state space $\Omega_1^d\times(\Omega_2^d\times\Omega_3)$, where $\Omega_1^d = \{\emptyset\}\cup\{J \mid$

$J = s_k \dots s_1, 0 \leq s_j \leq d, 1 \leq j \leq k, k \geq 1\}$, $\Omega_2^d = \{(m,i) \mid 0 \leq m \leq Q-3, -1 \leq i \leq d\} \cup \{(Q-2,i) \mid 0 \leq i \leq d\} \cup \{(Q-1,i) \mid 2 \leq i \leq d\}$ and $\Omega_3 = \{j \mid 1 \leq j \leq l\}$. The transition matrices D_k, U_s and $A_{k,s}$ are the following. The entries of the matrices $A_{k,s}$ are all zero. Thus, the chain does not allow transitions between sibling nodes.

The matrices D_k hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ goes from state $(J+k, ((m,i),j))$ to the state $(J, ((m',i'),j'))$. For $m=0$ and $i \neq -1$, we get

$$D_k(((0,i),j),((m',i'),j')) = \begin{cases} (B_{i'-k})_{j,j'} & m'=0, i \leq 1, i' \geq k, i' < d, \\ \sum_{l \geq d-k} (B_l)_{j,j'} & m'=0, i \leq 1, i' \geq k, i' = d, \\ 0 & \textit{otherwise}, \end{cases} \tag{4.15}$$

For $m=0$ and $i=-1$, we set

$$D_k(((0,-1),j),((m',i'),j')) = \begin{cases} (B_0)_{j,j'} & k = 0 \text{ or } 1, m'=0, i'=-1, \\ (B_{i'})_{j,j'} & k = 0 \text{ or } 1, m'=0, d > i' > 0, \\ \sum_{l \geq d} (B_l)_{j,j'} & k = 0 \text{ or } 1, m'=0, i'=d, \\ (I_l)_{j,j'} & k > 1, m' = Q-1, i' = k, \\ 0 & \textit{otherwise}, \end{cases} \tag{4.16}$$

where I_l is a $l \times l$ identity matrix. For $m \neq 0$, all entries of $D_k, 0 \leq k \leq d$, are zero.

The matrices U_s hold the transition probabilities that the chain $(\mathcal{X}_t^d, \mathcal{M}_t^d)$ goes from state $(J+k, ((m,i),j))$ to the state $(J+ks, ((m',i'),j'))$. For $m=0$ or $Q-1$, we get

$$U_s(((m,i),j),((m',i'),j')) = \begin{cases} C_s^i q_{Q-1}^s (1-q_{Q-1})^{i-s} (I_l)_{j,j'} & m' = Q-2, i > 1, \\ & i \geq s, i' = i-s, \\ 0 & \textit{otherwise}. \end{cases} \tag{4.17}$$

For $m=1$ and $i \geq 0$,

$$U_s(((1,i),j),((m',i'),j')) = \begin{cases} C_s^i q_1^s (1-q_1)^{i-s} (B_{i'-(i-s)})_{j,j'} & m'=0, i \geq s, \\ & d > i' \geq i-s, \\ C_s^i q_1^s (1-q_1)^{i-s} \sum_{l \geq d-(i-s)} (B_l)_{j,j'} & m'=0, i \geq s, i'=d, \\ 0 & \textit{otherwise}. \end{cases} \tag{4.18}$$

For $m=1$ and $i=-1$,

$$U_s(((1,-1),j),((m',i'),j')) = \begin{cases} (B_0)_{j,j'} & m'=s=0, i'=-1, \\ (B_{i'})_{j,j'} & m'=s=0, 0 < i' < d, \\ \sum_{l \geq d} (B_l)_{j,j'} & m'=s=0, i'=d, \\ 0 & \textit{otherwise}. \end{cases} \tag{4.19}$$

While, for $m = Q-2$,

$$U_s(((Q-2,i),j),((m',i'),j')) = \begin{cases} C^i_s q^s_{Q-2}(1-q_{Q-2})^{i-s}(I_l)_{j,j'} & m' = Q-3, i > 0, \\ & i \geq s, i' = i-s, \\ (I_l)_{j,j'} & m' = Q-3, \\ & i = s = 0, i' = -1, \\ 0 & \textit{otherwise}. \end{cases} \tag{4.20}$$

Finally, for $1 < m < Q-2$, we have

$$U_s(((m,i),j),((m',i'),j')) = \begin{cases} C^i_s q^s_m(1-q_m)^{i-s}(I_l)_{j,j'} & m' = m-1, i > -1, \\ & i \geq s, i' = i-s, \\ (I_l)_{j,j'} & m' = m-1, s = 0, i = i' = -1, \\ 0 & \textit{otherwise}. \end{cases} \tag{4.21}$$

4.2.5 Stability of a Tree Structured QBD Markov Chain

In Section 3.4 we argued that the stability of a tree structured QBD Markov chain that only allows transitions to parent or child nodes can be determined as follows. Define $V[0] = 0$ and use the recursion

$$V[N+1] = \sum_{s=0}^{d} U_s(I - V[N])^{-1}D_s, \tag{4.22}$$

to calculate $V[N]$. The Markov chain is recurrent if the matrices $G_s[N] = (I - V[N])^{-1}D_s$ converge to a set of stochastic matrices G_s; otherwise, we have a transient chain. The iterative formula (4.22) can be further optimized by making use of the structural properties of the matrices D_s, U_s and $V[N]$. For the basic and the modified binary CTM algorithm with free access, this optimization was limited to an acceleration of the product of $(I - V[N])^{-1}$ with the matrices D_s, where we made use of the fact that about 80 percent of the rows of D_s contain nothing but zeros. For higher splitting factors Q, this percentage is even higher (90 to 95 percent). The inversion of the matrix $I - V[N]$ was also optimized for $Q > 2$. We will demonstrate this for the basic CTM algorithm with $Q > 2$; the technique is similar (slightly more complex) for the modified scheme with $Q = 3$ and $Q > 3$.

Consider the $l(d+1)(Q-1) \times l(d+1)(Q-1)$ matrix V (see Section 3.4 for its definition), that corresponds to the tree structured QBD Markov chain presented in Section 4.2.1, the $(i,v)^{th}$ element of which is the taboo probability that starting from $(J+k,i)$, the chain eventually returns to a node with the same length as $J+k$ by visiting $(J+k,v)$, under the taboo of the node J and the sibling nodes of $J+k$. Next, subdivide the matrix V in blocks of size $l(d+1) \times l(d+1)$.

$$V = \begin{pmatrix} V_{0,0} & V_{0,1} & \dots & V_{0,Q-2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{Q-2,0} & V_{Q-2,1} & \dots & V_{Q-2,Q-2} \end{pmatrix}, \tag{4.23}$$

where the elements of V_{q_1,q_2} are the taboo probabilities that starting from $(J+k,(i,j,q_1))$, the chain $(\mathcal{X}_t^d, \mathcal{N}_t^d)$ with $\mathcal{N}_t^d = (\mathcal{Y}_t^d, \mathcal{Z}_t, \mathcal{Q}_t)$ eventually returns to a node with the same length as $J+k$ by visiting $(J+k,(v,u,q_2))$, under the taboo of the node J and the sibling nodes of $J+k$. Looking at the transition probabilities of $(\mathcal{X}_t^d, \mathcal{N}_t^d)$, these taboo probabilities are equal to zero if $q_2 \neq 0$. Thus,

$$V = \begin{pmatrix} V_{0,0} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ V_{Q-2,0} & 0 & \dots & 0 \end{pmatrix}. \tag{4.24}$$

The inverse $(I-V)^{-1}$ of a matrix V with such a structure is found as

$$(I-V)^{-1} = \begin{pmatrix} (I-V_{0,0})^{-1} & 0 & 0 & \dots & 0 \\ V_{1,0}(I-V_{0,0})^{-1} & I & 0 & \dots & 0 \\ V_{2,0}(I-V_{0,0})^{-1} & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ V_{Q-2,0}(I-V_{0,0})^{-1} & 0 & 0 & \dots & I \end{pmatrix}. \tag{4.25}$$

Clearly, the matrices $0 \leq V[N] \leq V, N \geq 0$, have the same structure as V and therefore, we can reduce the complexity of the matrix inversion in (4.22) from $O(l^3d^3Q^3)$ to $O(l^3d^3Q)$. Moreover, the structure of $V[N]$ also implies that only the first $l(d+1)$ columns of the matrix products between the matrices U_s and $(I-V[N])^{-1}D_s$ differ from zero, allowing us to reduce the complexity of these products from $O(l^3d^3Q^3)$ to $O(l^3d^3Q^2)$.

It is not too difficult to generalize the equations presented in Section 3.6. That is, many interesting performance measures—including the mean delay—can be calculated from the steady state probabilities of each of these Markov chains. Numerical results that compare the mean delay—and some other measures as well— for different values of Q are presented in the next section.

4.3 Numerical Results

We determine the instability point, i.e., maximum achievable throughput, of the basic and the modified CTM algorithm for different arrival processes that belong to the class of the D-BMAP processes. We mainly consider fair coins, i.e., $p_0 = p_1 = \ldots = p_{Q-1} = 1/Q$, and shortly discuss biased coins for $Q = 2$. The D-BMAP input processes considered were introduced in Section 2.1.3. We start with the results for the basic CTM algorithm with fair coins (for different values of Q). Some figures on the average delay and the expected number of retransmission are also presented.

4.3.1 The Basic CTM Algorithm with Fair Coins

Maximum Stable Throughput

Table 4.1 presents the stability points, i.e., maximum achievable throughput, of nine different arrival processes: the Poisson process, three Markov modulated Poisson processes, three Bulk arrival processes and two Erlang processes and this for $Q = 2, 3, 4$ and 5. For the Poisson process, resp. the Erlang processes, we start with $\lambda = 0$, resp. $\lambda_e = 0$, and increase λ, resp. λ_e, until instability is reached. For the bulk arrival processes we fix v and decrease L until instability is reached (we started with a large value of L). Finally, for the Markov modulated Poisson processes we fix a, b and λ_2 (the last one possibly as a function of λ_1) and increase λ_1 until instability is reached. For each couple (a, Q), where a is an arrival process and Q the splitting factor, Table 4.1 presents two values x and y. The first x is the lower bound α of the interval $]\alpha, \alpha + .001[$ that holds the instability point of the arrival process a, i.e., the maximum arrival rate λ of the D-BMAP for which it is stable. The second y indicates the difference between α and β in multiples of .001, where $]\beta, \beta + .001[$ holds the instability point of the Poisson process.

Let us study these results in detail. The Poisson results presented in Table 4.1 are in complete correspondence with the results obtained in [43]. This means that the results obtained by Mathys and Flajolet [43] lie within the intervals presented in Table 4.1. Replacing the input Poisson process by a Markov modulated Poisson process results in an inferior stability. This implies that more bursty and more correlated (compare the second MMPP with the third) input traffic results in a worse stability, i.e., a lower maximum achievable throughput. Moreover, the higher the splitting factor Q the larger the throughput degradation, e.g., replacing the Poisson input by $M(\cdot, 0, 30, 30)$ input results in a loss of .012 for $Q = 2$, .026 for $Q = 3$, .035 for $Q = 4$ and .041 for $Q = 5$. Therefore, lower splitting factors Q are better equipped to cope with bursty and correlated input traffic. Intuitively, one can understand this as follows. More bursty and correlated traffic generally results in more collisions. A collision results in an increment of the current stack level of all backlogged stations. The higher Q the higher the increment. Thus, for every collision one needs at least $Q - 1$ empty or successful slots in order to return to the same current stack level. Therefore, higher splitting factors suffer more under increased burstiness. Or to state it differently, the basic Q-ary CTM algorithm with free access is unstable if Q times the probability that a slot holds a collision is larger than one; whereas the number of initial collisions due to simultaneous new arrivals in a time slot are identical whichever splitting factor Q is being used.

Also, notice that a factor $Q = 2$ performs better, .004, than a factor $Q = 5$ for the $M(\cdot, 0, 300, 300)$ process (for Poisson input it was the opposite). As a matter of fact, for any two factors Q_1 and Q_2, within the range $[2, 5]$, one can always find an input process for which the factor Q_1 outperforms the factor Q_2, except for $Q_1 = 2$ and $Q_2 = 3$ (see Table 4.1).

Let us now consider the Erlang results. Replacing the input Poisson process by an Erlang process results in a superior stability. This result corresponds with the previous result, i.e., less bursty traffic results in a higher maximum achievable throughput. Moreover, the

Process	$Q = 2$		$Q = 3$		$Q = 4$		$Q = 5$	
$PP(\cdot)$	.360	+0	.401	+0	.399	+0	.387	+0
$M(\cdot, 2\lambda_1, 30, 30)$	.358	-2	.397	-4	.393	-6	.380	-8
$M(\cdot, 0, 30, 30)$	.348	-12	.375	-26	.364	-35	.346	-41
$M(\cdot, 0, 300, 300)$	.347	-13	.373	-28	.361	-38	.343	-44
$ER(\cdot, 2)$	.365	+5	.419	+18	.427	+28	.425	+38
$ER(\cdot, 3)$	.367	+7	.427	+26	.441	+42	.444	+57
$B([2], \cdot)$	.348	-12	.359	-42	.327	-72	.291	-96
$B([3], \cdot)$	.349	-11	.372	-29	.352	-47	.325	-62
$B([4], \cdot)$	.348	-12	.371	-30	.355	-44	.332	-55

Table 4.1: *Stability results for the basic Q-ary CTM algorithm with free access*

higher the splitting factor Q the larger the increment, e.g., replacing the Poisson input by $ER(\cdot, 3)$ input results in a gain of .007 for $Q = 2$, .026 for $Q = 3$, .042 for $Q = 4$ and .057 for $Q = 5$. Therefore, higher splitting factors Q are better equipped to take advantage of less bursty input traffic (the explanation is the same as before). Finally, the Bulk arrival processes—the most artificial of the processes considered—are mainly introduced to indicate that exotic arrival patterns can seriously deteriorate the stability of the basic CTM algorithm, especially for higher splitting factors Q. For the binary scheme the loss is only about .012; whereas for $Q = 5$ it varies between .055 and .096. If we were to increase Q even more, things only become worse, e.g., for $Q = 10$ the basic CTM algorithm with free access is unstable for an arrival rate $\lambda = .18$ under $B([2], \cdot)$ input traffic.

In conclusion, when implementing the basic CTM algorithm, one should always select a splitting factor $Q = 2$ or 3 because the throughput degradation due to the introduction of correlation and burstiness is less severe for a low splitting factor Q, e.g., the difference observed between the worst possible and the best input traffic is .02 for $Q = 2$ (see Table 4.1). Although, the basic ternary CTM algorithm is more sensitive to the specific nature of the input process, i.e., the variation of the maximum achievable throughput is higher compared to the binary scheme, it still remains a practical optimum because, for each of the nine processes considered, it outperforms the binary scheme.

There is another important conclusion that can be drawn from these results. In the previous chapter we did not manage to find a primitive D-BMAP with an arrival rate $\lambda < \ln(2)/2$ for which the basic binary CTM algorithm (with free access) is unstable, where $\ln(Q)/Q$ is the maximum stable throughput for the basic Q-ary CTM algorithm with blocked access (see Section 2.3). That is, for each of the arrival processes considered the basic binary CTM algorithm with free access outperformed its blocked access counterpart. Actually, we believe that this might be the case for all the arrival processes belonging to the class of primitive D-BMAPs. In this section we did however manage to find an arrival process, e.g., the $B([2], \cdot)$, for which the maximum stable throughput is below $\ln(Q)/Q$ for $Q = 3, 4$ and 5 (compare Tables 1.1 and 4.1). Thus, the basic Q-ary CTM algorithm with free access can be outperformed by its blocked access counterpart for $Q = 3, 4$ and 5. Moreover, we can easily prove the following theorem.

THEOREM **4.1** *For any integer value $Q > 2$, there exists a primitive D-BMAP $(B_n)_n$ with an arrival rate $\lambda < \ln(Q)/Q$ such that the basic Q-ary CTM algorithm with free access is unstable under $(B_n)_n$ input traffic.*

The proof for $Q = 3, 4$ and 5 is given by Table 4.1. Similarly, we found that the algorithm with $Q = 6$, resp. $Q = 7$, was unstable under $B([2], 5.78)$, resp. $B([2], 6.27)$, traffic. The arrival rate λ of these two arrival processes is $.295 < \ln(6)/6$ and $.275 < \ln(7)/7$. Thus, it suffices to prove the theorem for $Q > 7$. Let $(B_n)_n$ be an arbitrary D-BMAP with $B_1 = 0$, $B_2 \neq 0$ and $B_n = 0$ for $n > 2$, e.g., the $B([2], L)$ arrival process. Thus, all new arrivals occur in groups of two. As a result, the probability p_c that a collision occurs is at least $\lambda/2$ (if $\lambda < 2$). Now, looking at the Markov chain constructed to evaluate the basic Q-ary CTM algorithm with free access, it is clear that the basic Q-ary CTM algorithm is unstable whenever the probability p_c of having a collision is larger than $1/Q$. Indeed, the probability that a transition is made to a parent node $(1 - p_c)$ must be larger than $(Q - 1)$ times the probability p_c of making a transition to a child node in order to have stability because each collision causes the Markov chain to decrease $(Q - 1)$ levels. Thus, the scheme is unstable under $(B_n)_n$ traffic if $\lambda \geq 2/Q$. Furthermore, $\lambda = 2/Q < \ln(Q)/Q$ if $2 < \ln(Q)$, this is true for $Q > 7.39$. This completes the proof.

In conclusion, for $Q > 2$, there exists a D-BMAP for which the basic Q-ary CTM algorithm with blocked access outperforms its free access counterpart. Note however that the D-BMAPs used to prove the theorem are very artificial and have little or no practical relevance.

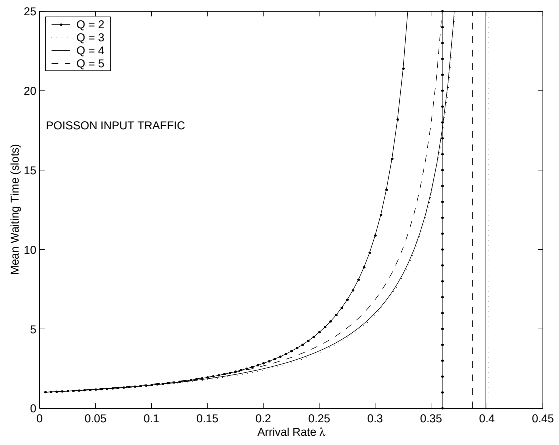


Figure 4.2: *The mean delay of the basic CTM algorithm with free access under Poisson input*

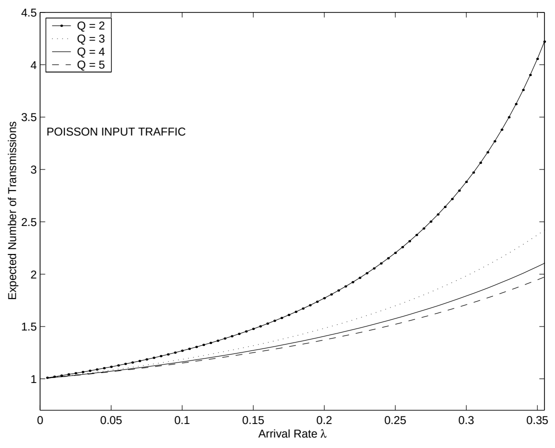


Figure 4.3: *The expected number of transmissions of the basic CTM algorithm under Poisson input*

Other Performance Measures

Figures 4.2 and 4.3 present the mean delay and the expected number of transmissions respectively as a function of the arrival rate λ under Poisson input. Figure 4.2 has often been used to indicate that having a higher stability point implies a better delay for every arrival rate λ below the maximum achievable throughput. This property is however not always valid for other arrival processes. For instance, Figure 4.5 clearly indicates that the expected delay for $Q = 5$ is (much) smaller than the mean delay for $Q = 2$ if $.2 < \lambda < .33$, whereas the binary scheme has a higher maximum stable throughput. Figure 4.4 presents the mean delay under $ER(\lambda_e, 3)$ traffic. Notice the big difference between the mean delay under Erlang, Poisson and Markov modulated Poisson traffic.

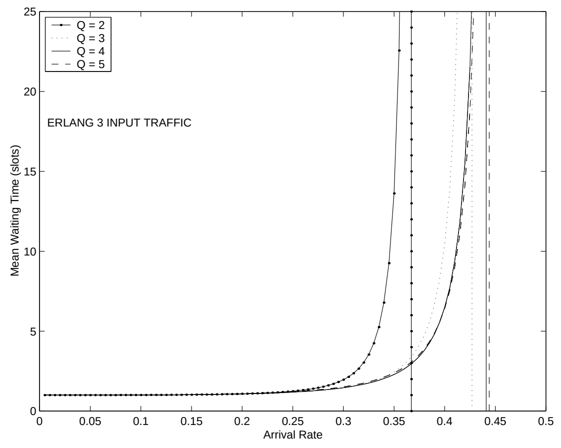


Figure 4.4: *The mean delay of the basic CTM algorithm with free access under* $ER(\lambda_e, 3)$ *input*

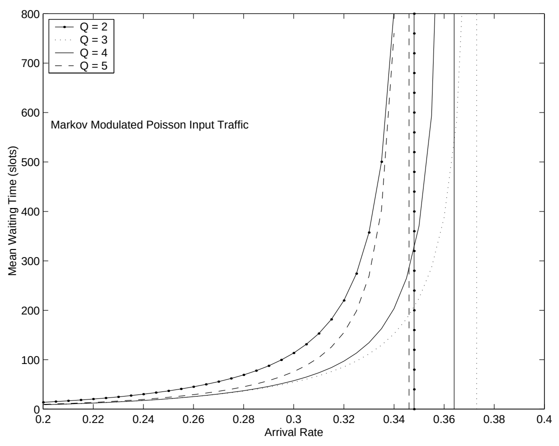


Figure 4.5: *The mean delay of the basic CTM algorithm with free access under* $M(\lambda_1, 0, 30, 30)$ *traffic*

Next, we investigate the influence of the correlation between the number of arrivals in consecutive time slots, on the mean delay and the expected number of transmissions. Consider the $M(\lambda_1, 0, 30, 30)$ arrival process. In order to study the influence of correlation we fix the arrival rate λ and gradually increase the mean sojourn time of both states (starting at $a = b = 30$).

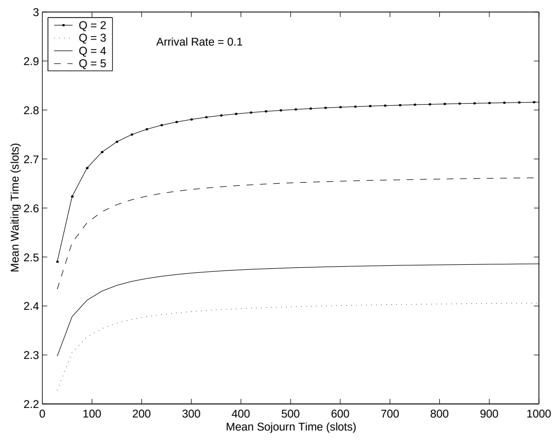


Figure 4.6: *The influence of correlation on the mean delay for* $\lambda = .1$

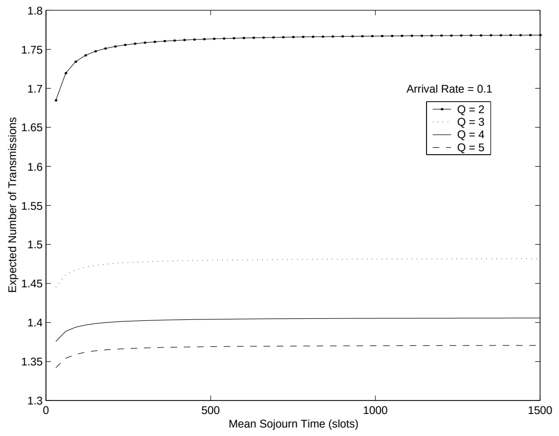


Figure 4.7: *The influence of correlation on the expected number of transmissions for* $\lambda = .1$

Figure 4.6 and 4.7 present the results for $\lambda = .1$; Figure 4.8 and 4.9 for $\lambda = .2$. Figure 4.6 and 4.7 indicate that the influence of correlation is hardly noticeable if the arrival rate is small. This is due to the fact that the mean arrival rate of both states is well below the maximum achievable throughput. On the other hand, Figure 4.8 and 4.9 indicate that the expected number of transmission remains small even under high correlation and high arrival rates; whereas the mean delay increases significantly as a result of the strong correlation. This strong increase follows from the fact that $\lambda_1 = .4$, while the maximum stable throughput of these processes is below .4 (see Table 4.1). Also, the ternary scheme

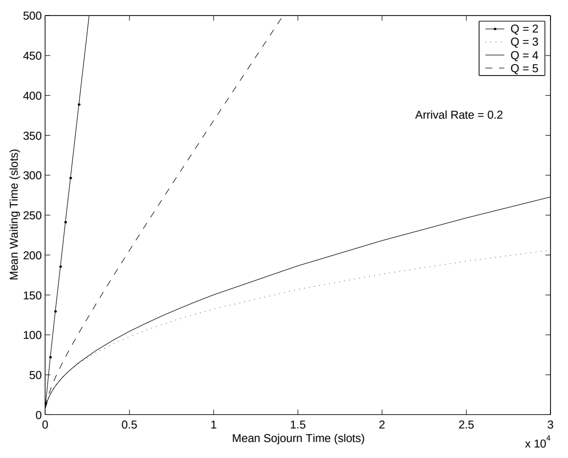


Figure 4.8: *The influence of correlation on the mean delay for $\lambda = .2$*

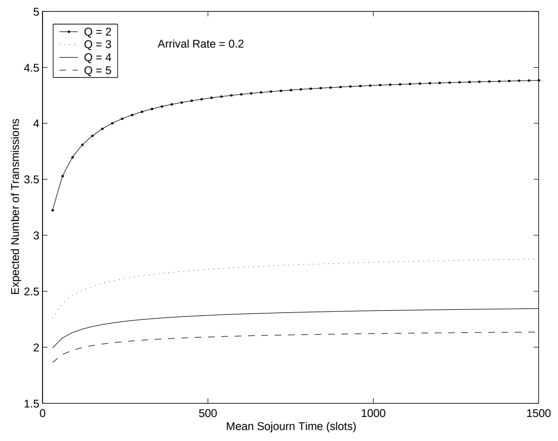


Figure 4.9: *The influence of correlation on the expected number of transmissions for $\lambda = .2$*

captures the influence of the correlation better than the other schemes. This comes as no surprise because the ternary scheme has the highest maximum stable throughput for this type of processes. In conclusion, the higher the maximum stable throughput of a scheme the better it copes with correlation.

4.3.2 The Modified CTM Algorithm with Fair Coins

Table 4.2 represents the stability results for the same nine arrival processes studied in the previous subsection. For each couple (a, Q), where a is an arrival process and Q the splitting factor, Table 4.2 presents two values x and y. The first x is the lower bound α of the interval $]\alpha, \alpha + .001[$ that holds the instability point of the arrival process a. The second y denotes the difference between the lower bounds α of the modified and the basic CTM algorithm (in multiples of .001).

The results for the Poisson process are in complete correspondence with the results obtained by Mathys and Flajolet [43]. When we focus on the result for $Q = 3$, we see that the Markov chain was unstable for an arrival rate of .407. Mathys and Flajolet [43] showed that the actual stability point is .40697 (see Table 1.1). This is another strong argument that the impact of the parameter d is indeed very small. Let us explain this in more detail. We know that instability of the approximated Markov chain always implies the instability of the exact Markov chain. The only possible error exists in the fact that the approximated chain might become stable when the exact chain is not. This might happen when we choose an arrival rate λ that is fractionally larger than the actual stability point.

Process	$Q = 2$		$Q = 3$		$Q = 4$		$Q = 5$	
$PP(\cdot)$	.388	+27	.406	+5	.400	+1	.387	+0
$M(\cdot, 2\lambda_1, 30, 30)$	.384	+26	.402	+5	.395	+2	.381	+1
$M(\cdot, 0, 30, 30)$	.371	+23	.380	+5	.365	+1	.346	+0
$M(\cdot, 0, 300, 300)$	.370	+23	.377	+4	.362	+1	.343	+0
$ER(\cdot, 2)$	.394	+29	.424	+5	.429	+2	.425	+0
$ER(\cdot, 3)$	.396	+29	.432	+5	.443	+2	.444	+0
$B([2], \cdot)$	.377	+29	.365	+6	.328	+1	.291	+0
$B([3], \cdot)$	.378	+29	.378	+6	.353	+1	.325	+0
$B([4], \cdot)$	.377	+29	.378	+7	.357	+2	.333	+1

Table 4.2: *Stability results for the modified Q-ary CTM algorithm with free access*

The result for $Q = 3$ shows that this is not the case even if the difference between both values, i.e., the arrival rate λ and the stability point, is only .00003.

Table 4.2 indicates that the impact of implementing the modified CTM algorithm is more or less the same for each of the arrival processes, e.g., for $Q = 2$ the increment varies between .023 and .027. Table 4.2 also confirms that it is hardly worthwhile to implement the modified CTM algorithm for $Q > 3$. The reason that *doomed* slots occur less frequent, for large Q, is twofold. First, the probability that all colliding stations select the last group is smaller (we use fair coins). Second, even if all colliding stations select the last group, a *doomed* slot only occurs if the next $Q - 1$ slots are unused by new arrivals. Table 4.2 indicates that there are arrival processes for which the modified binary CTM algorithm outperforms the ternary one, e.g., $B([2], \cdot)$.

As noted before, we did not manage to find a primitive D-BMAP with an arrival rate $\lambda < \ln(2)/2$ for which the basic binary CTM algorithm (with free access) is unstable, where $\ln(Q)/Q$ is the maximum stable throughput for the basic Q-ary CTM algorithm with blocked access (see Section 2.3). That is, for each of the arrival processes considered the basic binary CTM algorithm with free access outperformed its blocked access counterpart. For the modified binary CTM algorithm this is not the case. Indeed, the maximum stable throughput under $M(\cdot, 0, 300, 300)$ input is part of the interval $[.37, .371]$ for the modified binary CTM algorithm with free access, while its blocked access counterpart achieves a maximum stable throughput of .3754 under primitive D-BMAP input (see Section 2.3 and Table 1.1). Moreover, we have the following theorem, where $\ln(Q)/(Q - [Q^{-1} + (1 - Q^{-1})\ln(1 - Q^{-1})])$ is the maximum stable throughput of the blocked access algorithm:

THEOREM 4.2 *For any integer value $Q \geq 2$, there exists a primitive D-BMAP $(B_n)_n$ with an arrival rate $\lambda < \ln(Q)/(Q - [Q^{-1} + (1 - Q^{-1})\ln(1 - Q^{-1})])$ such that the modified Q-ary CTM algorithm with free access is unstable under $(B_n)_n$ input traffic.*

The proof for $Q = 2$ follows from the $M(\cdot, 0, 300, 300)$ result in Table 4.2; whereas the result for $Q = 3, 4$ and 5 follows from the $B([2], \cdot)$ result. For $Q = 6$ and 7 we made use of the $B([2], 5.78)$ and $B([2], 6.27)$ process respectively. Thus, it suffices to prove the

$PP(\cdot)$		$M(\cdot, 0, 30, 30)$		$ER(\cdot, 2)$	
p_0	$\alpha(d_s)$	p_0	$\alpha(d_s)$	p_0	$\alpha(d_s)$
.5000	.387	.5000	.371	.5000	.394
.4500	.391	.4500	.379	.4500	.397
.4300	.392	.4200	.382	.4400	.397
.4100	.393 (.0023)	.3800	.384 (.0018)	.4200	.398 (.0083)
.4068	.393 (.0024)	.3750	.384 (.0019)	.4175	.398 (.0084)
.4050	.393 (.0023)	.3700	.384 (.0018)	.4150	.398 (.0082)
.3800	.392	.3600	.384 (.0013)	.4100	.398 (.0073)
.3500	.390	.3400	.383	.3900	.397

Table 4.3: *Stability results for the modified binary CTM algorithm with free access and biased coins*

theorem for $Q > 7$. Now, $[Q^{-1} + (1 - Q^{-1}) \ln(1 - Q^{-1})]$ is positive for $Q > 1$. Hence, $\ln(Q)/Q$ is smaller than $\ln(Q)/(Q - [Q^{-1} + (1 - Q^{-1}) \ln(1 - Q^{-1})])$. Therefore, it suffices to prove that there exists a D-BMAP $(B_n)_n$ with an arrival rate $\lambda < \ln(Q)/Q$ such that we get instability. Let $(B_n)_n$ be an arbitrary D-BMAP with $B_1 = 0$, $B_2 \neq 0$ and $B_n = 0$ for $n > 2$, e.g., the $B([2], L)$ arrival process. Thus, all new arrivals occur in groups of two. As a result, the probability p_c that a collision occurs is at least $\lambda/2$ (if $\lambda < 2$). Now, looking at the Markov chain constructed to evaluate the modified Q-ary CTM algorithm with free access, it is clear that the algorithm becomes unstable whenever the probability p_c of having a collision is larger than $1/Q$. Indeed, the probability that a transition is made to a parent node $(1 - p_c)$ must be larger than $(Q - 1)$ times the probability p_c of making a transition to a child node in order to have stability because each collision causes the Markov chain to decrease $(Q - 1)$ levels. Thus, the modified Q-ary CTM algorithm is unstable under $(B_n)_n$ traffic if $\lambda \geq 2/Q$. Moreover, $\lambda = 2/Q < \ln(Q)/Q$ if $2 < \ln(Q)$, this is true for $Q > 7.39$. This completes the proof.

In conclusion, for $Q \geq 2$, there exists a D-BMAP for which the modified Q-ary CTM algorithm with blocked access outperforms its free access counterpart (see Equation 1.3 and Theorem 2.1). One must however note that the D-BMAPs used to prove the theorem are very artificial and have little practical relevance (except for the $Q = 2$ result).

4.3.3 Using Biased Coins

In Section 3.7.6 we discussed the use of biased coins when the basic binary CTM algorithm with free access is used. In this section we study the influence of biased coins for the modified binary CTM algorithm. In Section 3.7.6 we saw that the burstier the input traffic is the lower the optimal value of p_0 becomes whenever the basic binary CTM algorithm is used. Table 4.3 confirms that this is also the case for the modified binary CTM algorithm. However, for the modified algorithm the maximum stable throughput that can be achieved with biased coins differs much more from the maximum stable throughput achieved with fair coins (compared to the basic CTM algorithm, see Table 3.7). Moreover, the ranges of the optimal p_0's are very different from the ones that we found for the basic scheme (about .09 lower). This can be understood as follows: selecting a smaller value for p_0 becomes more attractive because a lower penalty is paid when all the colliding stations select the last (second) group compared to the basic CTM algorithm.

In conclusion, for bursty and correlated arrival patterns higher throughput results can be achieved by decreasing p_0, especially if the modified scheme is used. However, the optimal value for p_0 is hard to predict (it depends upon the stochastic nature of the arrival process).

4.4 Conclusions

We have analyzed the throughput characteristics of the basic and modified Q-ary CTM algorithm with free access for both fair and biased coins by constructing several tree structured QBD Markov chains and by determining their stability. As opposed to any prior work, we did not restrict our study to Poisson arrival patterns but considered a much more general class of input processes (D-BMAPs). We have shown, by means of numerical examples, that the binary and the ternary schemes should be preferred above higher splitting factors Q because they suffer much smaller throughput losses under bursty and correlated input traffic. The maximum stable throughput achieved by the binary and ternary CTM algorithm under D-BMAP input is not far below the Poisson result, i.e., the CTM algorithm with free access maintains its good stability characteristics under D-BMAP input. Moreover, whenever possible, it is worth to exploit ternary feedback, i.e., implement the modified scheme, for a splitting factor $Q = 2$ or 3. We also demonstrated that it might be very useful to use biased coins when the input traffic is expected to be highly bursty and correlated. Decreasing the probability that a station selects the first group (after a collision) results in higher throughput results.

If we compare the blocked access strategy with the free access scheme, we have proven (see Theorems 4.1 and 4.2) that there exists a primitive D-BMAP for which the basic and modified Q-ary CTM algorithm with blocked access outperforms its free access counterpart (except for the basic binary CTM algorithm). The D-BMAPs used to prove these theorems are however of a rather artificial nature and therefore of lesser practical importance. For those D-BMAPs that are of a more practical nature, we may conclude that free access generally results in (slightly) better throughput.

Another important performance characteristic is the mean delay that is experienced when transmitting a packet. Using the QBD Markov chains that were constructed in this thesis, it is possible to calculate the mean delay and many other performance characteristics. Numerical results have indicated that a higher maximum stable throughput does not necessarily imply a smaller delay for every arrival rate λ. This was a hope expressed by many researchers, e.g., Massey [42] who states "If one algorithm has a larger maximum stable throughput than another, one hopes that if the first algorithm is reasonably simple (so that the large maximum stable throughput was not achieved by "trickery" that used high arrivals rates to a special advantage) then the first algorithm will have a better delay-throughput characteristic for all throughputs." Notice, the delay curves for the Poisson input seemed to confirm this hope, but the $M(\cdot, 0, 30, 30)$ input indicated that this is not always the case. Nevertheless, if an algorithm has a larger maximum stable throughput, it is expected to cope better with correlation.

The Bit Error Ratio (BER) and capture effects are important characteristics of a wireless channel. It is fairly straightforward to see that one can extend the models presented in this thesis in order to evaluate the CTM algorithm with free access when applied to a channel with Markovian capture and errors. For instance, one could easily add the state of the channel as a part of the auxiliary variable of the tree structured QBD Markov chains.

Chapter 5

Tree Algorithms and Grouping

In this chapter we investigate the stability of tree algorithms that make use of a grouping strategy. A number of tree algorithms of this type were introduced in Section 1.4.4. We do not consider Gallager's optimized version that uses the arrival times to split colliding stations into two groups (the discrete nature of the D-BMAP arrival process prohibits us from doing so). The two other algorithms discussed in Section 1.4.4 are introduced again in the next section and their stability under D-BMAP traffic is discussed in Section 5.2 and 5.3. Conclusions are drawn in Section 5.4.

5.1 Tree algorithms using a Grouping Strategy

A description of the grouping mechanism due to Massey [41] is given below. Suppose that the random access scheme is activated at time $t = 0$. The unit of time is defined as the length of a slot, so that the i-th transmission slot is the time interval $(i, i + 1]$. A second time increment Δ is chosen and the i-th arrival epoch is defined as the time interval $(i\Delta, i\Delta+\Delta]$ (Δ is not necessarily an integer value). The first transmission rule used by this algorithm is as follows: transmit a new packet that arrived during the i-th arrival epoch in the first utilizable slot following the collision resolution interval (CRI) for new packets that arrived during the $(i - 1)$-th arrival epoch. The modifier "utilizable" reflects the fact that the CRI for new packets that arrived during the $(i - 1)$-th arrival epoch might end before the i-th arrival epoch has ended. If so, a number of transmission slots are skipped until the i-th arrival epoch ends. One could improve the algorithm by shortening the i-th arrival epoch. This both complicates the analysis and the implementation and is expected to have no influence on the maximum stable throughput (because it only alters the behavior of the algorithm when there are no backlogged groups).

Each of the groups is resolved using either the basic binary or the modified binary CTM algorithm, depending on whether we have binary or ternary feedback (the order in which the groups are resolved is of no importance). The CTM algorithm with a higher splitting factor $Q > 2$ is not expected to improve the maximum stable throughput if Δ is small (see Section 1.4.4). When a grouping strategy is being used, both active and inactive stations have to monitor the channel continuously (this is also true for algorithms that apply a blocked access strategy).

5.2 Stability under D-BMAP Traffic

It is not too difficult to determine the maximum stable throughput of the two algorithms introduced in Section 5.1. We restrict ourselves to the case where Δ, the grouping interval, is an integer value. In Section 5.2.1 we prove that an algorithm that resolves the collisions using a grouping strategy is stable under primitive D-BMAP traffic if the expected time to resolve an arbitrary group $E[G]$ is smaller than Δ and unstable if $E[G] > \Delta$. Afterwards we indicate how to obtain tight upper and lower bounds on $E[G]$. For the two algorithms introduced in Section 5.1, these bounds allow us to determine the maximum stable throughput with sufficient accuracy.

5.2.1 A stability Condition for D-BMAP Input

An algorithm that applies a grouping strategy under primitive D-BMAP input traffic can be seen as a queue with the following characteristics. Assume that Δ is an integer. The customers arriving in the queue correspond to the groups produced by the algorithm. Thus, every Δ time slots a new customer arrives—that is, we have a deterministic arrival process. The queue has an infinite waiting room and a single server. A customer is said to be of type j with $1 \leq j \leq l$ if the state of the D-BMAP $(B_n)_n$ at the start of the corresponding grouping interval was j. The group types are therefore determined by a primitive discrete time Markov chain with transition matrix B^{Δ}, where B is the transition matrix of the D-BMAP $(B_n)_n$, i.e., $B = \sum_n B_n$. Thus, if the type of customer n is i than the type of customer $n+1$ is j with probability $(B^{\Delta})_{i,j}$. The service time of a customer—that is, the time required to resolve the corresponding group—depends upon the type of the customer. Thus, the service time of a customer of type j is t with some probability $G_j(t)$. Remark that the service time of a customer depends on the state of the D-BMAP at the start of the corresponding grouping interval. For l the number of states of the D-BMAP, or else the number of customer types, equal to one the above-mentioned queue reduces to a $D/G/1$ queue and such a queue is known to be stable for $\rho < 1$ [23]. This condition is obviously equivalent to $E[G] < \Delta$. Another way to prove that $E[G] < \Delta$ is a sufficient condition for stability when $l = 1$ is to use the Stability Lemma of Pakes [3, p264]. For $l > 1$, things are slightly more complicated.

The arrival process of our queue can be seen as a special case of the discrete time version of a Markovian arrival process with marked arrivals [25, 27], denoted as $MMAP[K]$. Such a Markov arrival process is characterized by a set of $m \times m$ matrices M_0 and M_J with J a string of integers, where each integer is part of $[1, K]$. The i, j-th element of M_J, with $J = j_1 \dots j_n$, $n > 0$, represents the probability that a transition is made from state i to j and that n arrivals occur. The type of these n arrivals is as follows: the k-th customer that arrives is a customer of type j_k. The matrix M_0 characterizes the transitions when no new arrivals occur. For $K = 1$ the $MMAP[K]$ arrival process reduces to a D-BMAP arrival process (if we identify the matrix B_n with M_J where J is a string that consists of n ones). It is easily seen that the arrival process of our queue of interest is actually a $MMAP[K]$ process with $K = l$ and $m = \Delta l$. The matrix M_0 has the following form:

$$M_0 = \begin{pmatrix} 0 & I & 0 & \dots & 0 & 0 \\ 0 & 0 & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I & 0 \\ 0 & 0 & 0 & \dots & 0 & I \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}, \tag{5.1}$$

where I is the $l \times l$ unity matrix. The matrices M_k, $1 \leq k \leq l$, obey the following equation:

$$M_k = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ B^{\Delta}(k) & 0 & 0 & \dots & 0 & 0 \end{pmatrix}, \tag{5.2}$$

where $B^{\Delta}(k)$ is obtained from B^{Δ} by keeping the k-th column of the matrix B^{Δ} and setting all other elements to zero. The entries of the matrices M_J with J a string of length 2 or more are all zero. Now that we know that the input is a $MMAP[K]$, the queue we are interested in is a special case of a MMAP[K]/G[K]/1 queue.

He [25] has shown that a MMAP[K]/G[K]/1 queue with a work conserving service discipline is positive recurrent if and only if $\rho = \lambda_1 E[G_1] + \ldots + \lambda_K E[G_K] < 1$ and it is transient if $\rho > 1$, where λ_i corresponds to the average number of type i customers arriving in the queueing system (per time unit) and $E[G_i]$ to the expected service time of a type i customer. In our case the vector $(\lambda_1, \ldots, \lambda_K)$ is nothing but β/Δ, where $\beta B = \beta$ and $\beta e = 1$ (because β is also the invariant vector of B^{Δ}). Thus, $\Delta\rho$ is equal to the expected service time of an arbitrary customer—that is, the expected time required to resolve an arbitrary group. This proves that we get a stable, resp. unstable, system whenever $E[G] < \Delta$, resp. $E[G] > \Delta$.

5.2.2 Tight Bounds on $E[G]$

Following Massey's approach [41] it is fairly straightforward to obtain a tight upper and lower bound on $E[G]$ when the basic or modified binary CTM algorithm is used to resolve the groups. First, we determine the probability that a group contains n contenders—that is, n arrivals occur in the corresponding interval of length Δ. The probability that the state of the D-BMAP is j, $1 \leq j \leq l$, at the start of a grouping interval is equal to β_j, where β_j is the j-th component of the stationary vector β corresponding to the D-BMAP $(B_n)_n$ because β is also an invariant vector of B^{Δ}. The probability of having n arrivals in an interval of length Δ provided that the state is j at the start of the interval, say $P_j(n)$, is easily computed as follows. Define the matrices $B_{n,i}, i > 1, n \geq 0$, as

$$B_{n,i} = \sum_{j=0}^{n} B_{j,i-1} B_{n-j}, \tag{5.3}$$

with $B_{n,1}$ equal to B_n. Then, $P_j(n)$ is found as the j-th component of $B_{n,\Delta}e$. Therefore, the probability that a group contains n arrivals, say $P(n)$, is nothing but $\sum_{j=1}^{l} \beta_j P_j(n)$.

The expected time required to resolve an arbitrary group $E[G]$ is found as $E[G] = \sum_n P(n)L(n)$, where $L(n)$ represents the expected time required by the collision resolution algorithm to resolve a set of n contenders. Massey [41] obtained the following upper and lower bounds on $L(n)$ for the basic and modified binary CTM algorithm. In order to distinguish both algorithms we write $L_b(n)$ for the expected time required by the basic binary CTM algorithm and $L_m(n)$ as the expected time required by the modified binary CTM algorithm. For the basic binary CTM algorithm we have

$$L_b(n) \leq a_1 n - 1 + 2\delta_{0,n} + (2 - a_1)\delta_{1,n} + (6 - 2a_1)\delta_{2,n} + (26/3 - 3a_1)\delta_{3,n}, \tag{5.4}$$

with $a_1 = 2.8867$ and $\delta_{i,j} = 0$ if $i \neq j$ and 1 if $i = j$. Moreover,

$$L_b(n) \geq a_2 n - 1 + 2\delta_{0,n} + (2 - a_2)\delta_{1,n} + (6 - 2a_2)\delta_{2,n} + (26/3 - 3a_2)\delta_{3,n}, \tag{5.5}$$

with $a_2 = 2.8810$. Whereas for the modified binary CTM we find

$$L_m(n) \leq b_1 n - 1 + 2\delta_{0,n} + (2 - b_1)\delta_{1,n} + (11/2 - 2b_1)\delta_{2,n} + (8 - 3b_1)\delta_{3,n}, \tag{5.6}$$

with $b_1 = 2.6651$ and

$$L_m(n) \geq b_2 n - 1 + 2\delta_{0,n} + (2 - b_2)\delta_{1,n} + (11/2 - 2b_2)\delta_{2,n} + (8 - 3b_2)\delta_{3,n}, \tag{5.7}$$

with $b_2 = 2.6607$. If we calculate $E[G] = \sum_n P(n)L(n)$ and replace $L(n)$ by its lower, resp. upper, bound we obtain a lower, resp. upper, bound on $E[G]$. Whenever the lower bound is larger than Δ we know from Section 5.2.1 that the algorithm is unstable, whereas if the upper bound is smaller than Δ we have a stable scheme. For those arrival rates that produce an upper bound larger than Δ and a lower bound that is smaller we know nothing. This procedure allows us to determine the stability point for any value of Δ with a precision of .001 or better ; that is, we can find an interval $[x, x + .001]$ that contains the maximum stable throughput of the algorithm.

5.3 Numerical Results

Before we present some actual numerical results, it is worthwhile to have a closer look at the upper and lower bounds of $L_b(n)$ and $L_m(n)$ presented in Section 5.2.2. With these bounds one can easily obtain an interval for each value of Δ that contains the maximum stable throughput under any primitive D-BMAP input traffic. The length of this interval will reduce as Δ is increased.

5.3.1 Selecting a Large Grouping Interval Δ

Using Equation (5.4) and $L_b(n) \leq a_1 n$ for $n > 0$, we have

$$\begin{aligned} E[G] &= \sum_{n \geq 0} P(n)L(n) \\ &\leq \sum_{n>0} a_1 n P(n) + P(0) \\ &= a_1 \lambda \Delta + P(0). \end{aligned}$$

Hence,

$$\lambda < \frac{1}{a_1}(1 - P(0)/\Delta), \tag{5.8}$$

is a sufficient condition for the stability of the grouping algorithm which uses the basic binary CTM algorithm to resolve the groups. Thus, only the presence of empty groups might reduce the maximum stable throughput below $1/a_1 = .3464$. This is a first indication that a grouping algorithm might not be able to support a high maximum stable throughput under bursty input traffic—that is, traffic of which the arrivals are concentrated in a small portion of the grouping intervals of length Δ. Numerical examples that confirm this idea are presented further on. Obviously, $P(0) < 1$ if $\lambda > 0$. As a result we have stability if

$$\lambda < \frac{1}{a_1}(1 - 1/\Delta), \tag{5.9}$$

for any primitive D-BMAP input traffic.

Using Equation (5.5), we have

$$\begin{aligned} E[G] &= \sum_{n} P(n)L(n) \\ &\geq a_2 \sum_n nP(n) - \sum_n P(n) + \\ &\quad 2P(0) + (2 - a_2)P(1) + (6 - 2a_2)P(2) + (26/3 - 3a_2)P(3) \\ &\geq a_2 \lambda \Delta - 1 + 2 - a_2 \\ &= a_2 \lambda \Delta - (a_2 - 1). \end{aligned}$$

Thus, the grouping algorithm that uses the basic binary CTM algorithm is unstable if

$$\lambda > \frac{1}{a_2}\left(1 + \frac{a_2 - 1}{\Delta}\right). \tag{5.10}$$

In conclusion, the maximum stable throughput of the grouping algorithm that uses the basic binary CTM algorithm to resolve the groups is found in the interval $[1/a_1(1 -$

Δ	*basic binary*		*modified binary*	
2	.1732	.6736	.1876	.6879
3	.2309	.5647	.2501	.5839
4	.2598	.5103	.2814	.5319
5	.2771	.4777	.3002	.5007
10	.3118	.4124	.3377	.4383
20	.3291	.3797	.3565	.4070
50	.3395	.3602	.3677	.3883
100	.3430	.3536	.3715	.3821
1000	.3461	.3478	.3748	.3765
10000	.3464	.3472	.3752	.3759
∞	.3464	.3471	.3752	.3758

Table 5.1: *Maximum achievable throughput for the basic and modified binary CTM algorithm when combined with a grouping strategy (fair coins)*

$1/\Delta), 1/a_2(1 + (a_2 - 1)/\Delta)]$. In other words, the algorithm is stable under primitive D-BMAP input traffic if $\lambda < 1/a_1(1 - 1/\Delta)$ and unstable if $\lambda > 1/a_2(1 + (a_2 - 1)/\Delta)$. Similarly, for the modified binary CTM algorithm we find the interval $[1/b_1(1 - 1/\Delta), 1/b_2(1 + (b_2 - 1)/\Delta)]$. Numerical results for different values of Δ are presented in Table 5.1. For instance, whatever the D-BMAP input processes might be its corresponding maximum stable throughput is found in the interval $[.3291, .3797]$, resp. $[.3565, .407]$ if $\Delta = 20$. In the next section we indicate that we can actually find arrival processes for which the maximum stable throughput is close to $1/a_1(1 - 1/\Delta)$ and $1/a_2(1 + (a_2 - 1)/\Delta)$. Hence, it is not possible to further reduce the size of the intervals in Table 5.1.

Obviously, for Δ large we find that the interval reduces to $[1/a_1, 1/a_2]$, resp. $[1/b_1, 1/b_2]$. Both these intervals are rather small and contain the maximum stable throughput of the corresponding algorithm with blocked access (see Section 1.4.4 and Section 2.3). Thus, whether the basic, resp. modified, binary CTM algorithm uses a blocked access strategy or a grouping strategy (with Δ large) makes little difference as far as the maximum stable throughput under primitive D-BMAP input traffic is concerned. In the next section we investigate what happens if Δ is small.

5.3.2 Selecting a Small Grouping Interval Δ

In this section we study the maximum achievable throughput as a function of Δ for different arrival processes. We subsequently discuss the discrete time Poisson process, Erlang processes, Markov Modulated Poisson processes and Bulk arrival processes. Definitions and abbreviations for these processes can be found in Section 2.1.3.

Markov Modulated Poisson Processes: We start with a discussion of the Markov modulated Poisson processes (MMPPs). Figure 5.1, resp. 5.2, compares the maximum stable throughput as a function of Δ ($2 \leq \Delta \leq 10$) for a few MMPPs when the basic, resp. modified, binary CTM algorithm is combined with a grouping strategy. Both figures are almost identical, except that the modified scheme supports throughputs which are a few percentages higher.

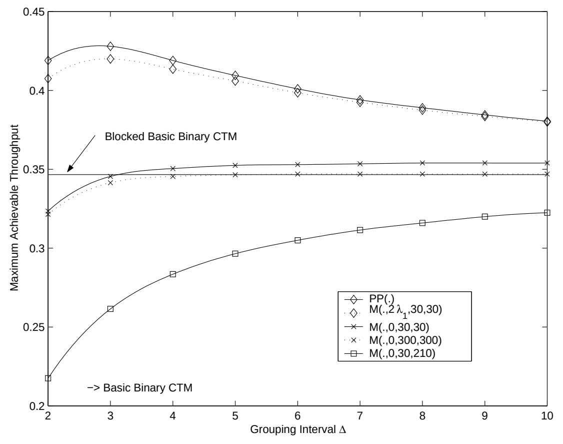


Figure 5.1: *The impact of Δ on the maximum stable throughput (basic)*

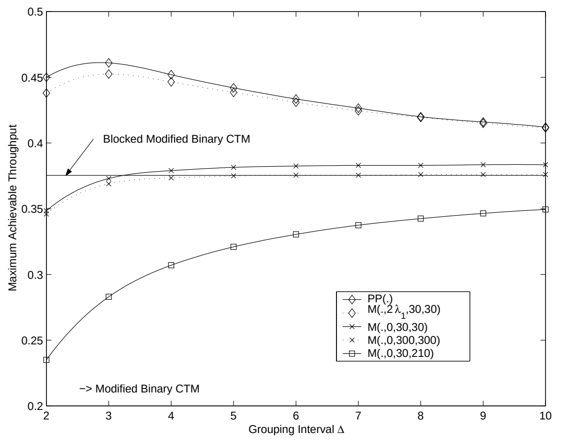


Figure 5.2: *The impact of Δ on the maximum stable throughput (modified)*

A first conclusion that can be drawn from both figures is that a serious degradation of the maximum stable throughput might occur if the burstiness (for a definition, see Section 2.1.2) of the arrival processes increases, especially if Δ is very small. The reason for this is the presence of the empty groups, as indicated in Section 5.3.1. Although the probability $P(0)$ of having an empty group does not decrease that rapidly when increasing Δ, the throughput degradation does disappear rather quickly. This is due to the fact that the throughput is actually a weighted sum of the throughputs T_i associated with a collision resolution interval (CRI) that corresponds with an interval that starts in state i. We refer to such a CRI as a type i CRI. The weight that corresponds to T_i depends upon the expected time necessary to resolve a type i CRI divided by sum over j of the expected time required to resolve a type j CRI. In the case of our $M(\cdot, 0, a, b)$ processes, we find that expected number of contenders associated with a type 1 CRI increases rapidly as Δ increases. Whereas the expected number of contenders in a type 2 CRI remains close to zero (for $\Delta << b$). This implies that the weight associated with $T_2 \approx 0$ decreases rapidly when Δ increases, which explains the rapid restoration of the maximum stable throughput when Δ is increased.

On the other hand, Figures 5.1 and 5.2 indicate that correlation is of lesser importance. For instance, the $M(\cdot, 0, 30, 30)$, the correlation function $r(k)$ of which decays as $.9333^k$, performs only slightly better than the $M(\cdot, 0, 300, 300)$, which has a correlation function $r(k)$ that decays as $.9933^k$. Moreover, the results for the $M(\cdot, 0, 3000, 3000)$ arrival process, which are not included in the figures, are almost identical to those of the $M(\cdot, 0, 300, 300)$ process. This comes as no surprise because the grouping mechanism breaks the correlation (i.e., the order in which the groups are resolved is of no importance).

Notice, the maximum stable throughput under $M(\cdot, 0, 30, 210)$ input traffic is only a few percentages higher than $1/a_1(1 - 1/\Delta)$, resp. $1/b_1(1 - 1/\Delta)$ (see Table 5.1). We can easily define a D-BMAP for which the maximum stable throughput is even closer to

$1/a_1(1-1/\Delta)$, resp $1/b_1(1-1/\Delta)$. For instance, the basic, resp. modified, CTM algorithm with grouping has a maximum stable throughput under $M(\cdot, 0, 30, 3000)$ input traffic of $\approx .1770$, resp. $\approx .1915$. The $M(\cdot, 0, 30, 3000)$ process is very bursty: the average sojourn time in the silent state is 3000 slots, whereas the average time in the active state is only 30 slots. Therefore, all the traffic is more or less concentrated in 1 percent of the grouping intervals of length Δ.

Erlang Arrival Process: Figures 5.3 and 5.4 present the results for the Erlang arrival processes. As expected we get a higher maximum stable throughput if k is increased, i.e., if the process becomes more deterministic. Also, the results for the $ER(\cdot, 10)$ process are only a few percentages below $1/a_2(1 + (a_2 - 1)/\Delta)$, resp. $1/b_2(1 + (b_2 - 1)/\Delta)$. For $k = 50$ we found a maximum stable throughput for $\Delta = 2$ of .623, resp. .6415. It is easy to prove that the maximum stable throughput for $\Delta = 2$ converges to .625, resp. .6429, as k approaches infinity.

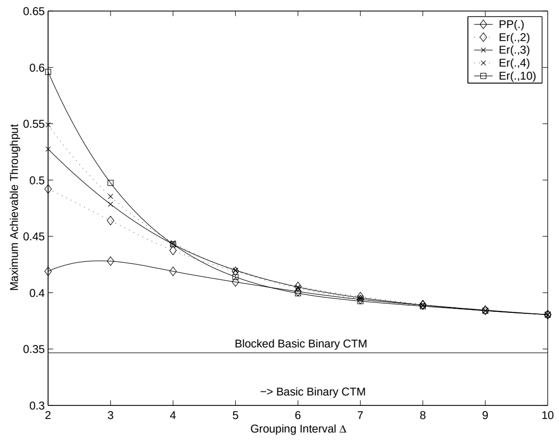


Figure 5.3: *The impact of Δ on the maximum stable throughput (basic)*

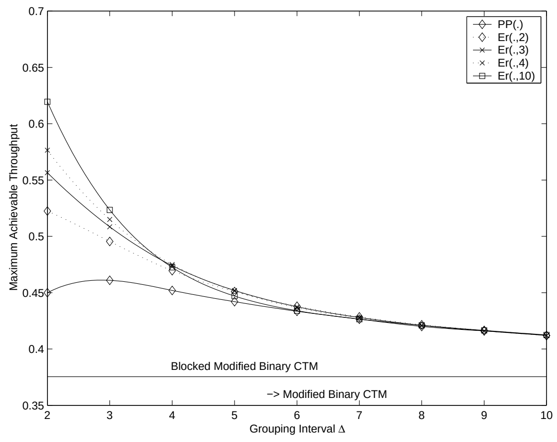


Figure 5.4: *The impact of Δ on the maximum stable throughput (modified)*

It is possible to find D-BMAP arrival processes for which these grouping algorithms support a higher maximum stable throughput (up to $1/a_2(1+(a_2-1)/\Delta)$, resp. $1/b_2(1+(b_2-1)/\Delta)$). For instance, the following primitive D-BMAP arrival process has a maximum stable throughput for $\Delta = 2$ of $\approx .6725$, resp. $\approx .687$.

$$B_0 = \begin{pmatrix} 0 & 0 & 0 \\ 1-1/p & 0 & 1/p \\ 0 & 0 & 0 \end{pmatrix}, B_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, B_{100} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

The other B_n matrices are zero. This arrival process was constructed such that $P(1) = 1-(x+y)$, $P(100) = x$ and $P(101) = y$ for $x+y$ small. The arrival rate $\lambda = (1+100/p)/(2+1/p)$. For p large, $\lambda \approx 1/2$ and both algorithms are stable (for $\Delta = 2$) when this D-BMAP is used as input traffic. In order to determine the maximum stable throughput, we decrease p, i.e., increase λ, until both grouping algorithms become unstable. Similar arrival processes can be constructed for $\Delta > 2$.

Bulk Arrival Process: Figures 5.5 and 5.6 present the results for some Bulk arrival processes. The results are in agreement with the explanations given in the MMPPs section.

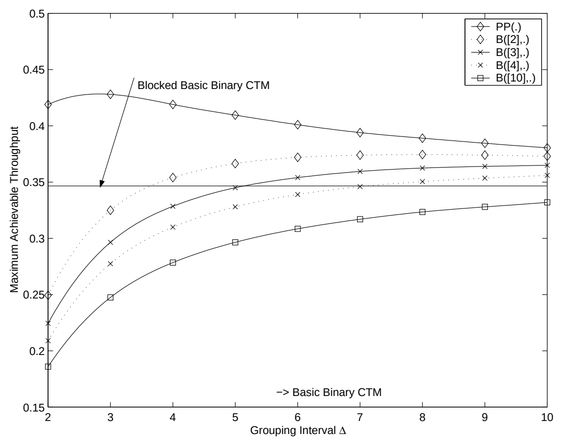


Figure 5.5: *The impact of Δ on the maximum stable throughput (basic)*

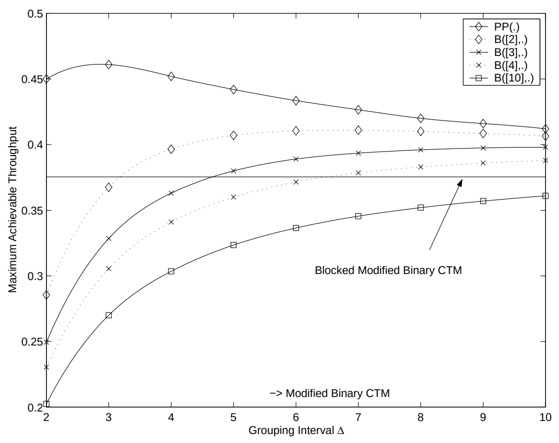


Figure 5.6: *The impact of Δ on the maximum stable throughput (modified)*

5.4 Conclusions

In this chapter we evaluated the stability of the basic and the modified binary CTM algorithm when combined with a grouping strategy under primitive D-BMAP traffic. The length of the grouping interval was denoted as Δ. We have proven that the basic scheme is stable under primitive D-BMAP traffic if the arrival rate $\lambda < 0.3464(1-1/\Delta)$ and unstable if $\lambda > 0.3471(1+1.881/\Delta)$, numerical values for these bounds are found in Table 5.1. A similar result was obtained for the modified scheme. These results imply that the grouping strategy provides similar stability guarantees as the blocked access strategy provided that Δ is chosen sufficiently large. Moreover, for small values of Δ one can find D-BMAPs with an arrival rate close to $0.3464(1-1/\Delta)$, resp. $0.3471(1+1.881/\Delta)$, that result in an unstable, resp. stable, behavior. In general, one may conclude that more bursty arrival processes have a smaller maximum stable throughput compared to the more deterministic ones (for small values of Δ).

Part II

Tree Algorithms in Wireless Access Networks

Chapter 6

The Identifier Splitting Algorithm combined with Polling (ISAP)

In this chapter the Identifier Splitting Algorithm combined with Polling (ISAP) is introduced. The influence of the different protocol parameters on the performance measures is studied in Chapter 7 by means of several analytical models. Numerical results are presented in Chapter 8. The Identifier Splitting Algorithm (ISA) was first introduced by Petras, *et al* [50–52] during the European RACE project 2067 on Mobile Broadband Systems (MBS) [44]. ISA is an algorithm used to resolve collisions occurring on the contention channel, present in the Medium Access Control (MAC) layer of the MBS protocol stack. The contention channel is used by the Mobile Stations to inform the Base Station about their current bandwidth requirements. The ISA scheme is a variation on the deterministic splitting algorithm introduced by Capetanakis [7, 17]. As opposed to the Capetanakis scheme, which traverses the contention tree in a depth-first approach, ISA uses a breadth-first approach.

ISA was designed to cope with the delayed feedback environment typically found in a wireless access network (see Section 6.1), whereas most splitting algorithms require immediate feedback due to the depth-first approach. Perhaps the most important advantages of the ISA scheme, or any other deterministic splitting algorithm, are the obvious upperbound provided on the worst case delay and the fact that splitting algorithms are known to perform well under low and high load conditions.

As a part of the European ACTS program a trial platform for Mobile Broadband Systems (MBS) was designed and implemented in the context of the SAMBA project (AC204) [57]. The trial platform used slotted ALOHA [1, 3] as the contention algorithm. This was not due to the fact that people had second thoughts about ISA, but simply because the trial platform consisted of 2 Base Stations and 2 Mobile Stations. Clearly, you do not need to implement a powerful contention resolution algorithm in an environment with only two competing Mobile Stations. If the number of Mobile Stations increases, random access becomes more important and more advanced collision resolution algorithms will be used to improve the performance of random access channels [54, 6.2: General Guidelines]. The purpose of this chapter is to introduce such an advanced contention resolution algorithm.

This chapter is structured as follows. In the next section, we introduce the concept of a delayed feedback environment. We proceed with the ISA protocol proposed by Petras, *et al* [50–52]. Next, we indicate how ISA can be combined with Polling, this combination is called the ISAP scheme. Afterwards, a number of optimizations are discussed. Finally, a flowchart for an MS using ISAP is presented.

6.1 A Delayed Feedback Environment

In this section we describe a framework for centralized wireless access networks. A number of MAC proposals found in literature fit into this framework: DSA++ [52, 74], D^{2}MA [37], EC-MAC [59] and [70–72].

Consider a cellular access network with a centralized architecture, i.e., the area covered by the wireless network is subdivided into a set of geographically distinct cells, each with a diameter of approximately 100m (slight overlaps are allowed to facilitate the handovers from one cell to a neighboring cell). Each cell contains a Base Station (BS) serving a finite set of Mobile Stations (MS). The MSs communicate among each other and with the nodes in the fixed network via the BS (see Figure 6.1).

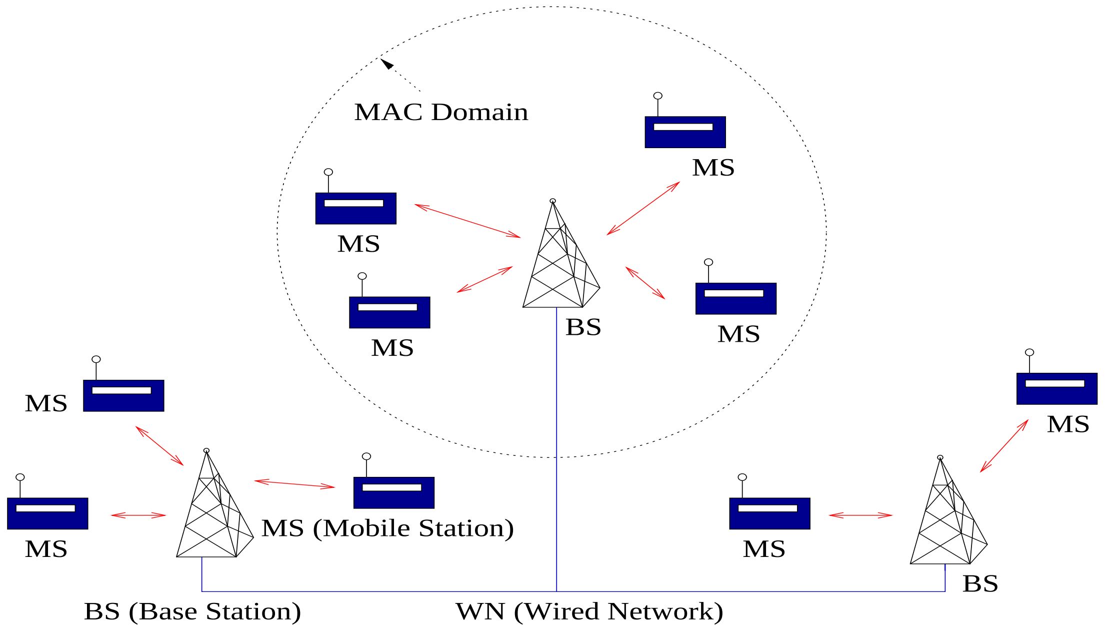


Figure 6.1: *Reference configuration of the system*

Two logically distinct communication channels (uplink and downlink) are used to support the information exchange between the BS and the MSs. Packets arriving at the BS are broadcasted downlink, while upstream packets must share the radio medium. The BS controls the access to the shared radio channel (uplink). The access technique used is Time Division Multiple Access (TDMA) combined with Frequency Division Duplex (FDD) to separate the uplink and downlink channels. The ISAP algorithm can also be implemented if the access technique is Time Division Duplex (TDD). In the further description we assume that we are working with an FDD system.

The battery consumption of a mobile node is still one of the main concerns when designing a wireless network [59]. To indicate the importance of power consumption: one of the main recommendations for future trial platforms made during the European SAMBA trial platform is to further reduce the size and power consumption of Mobile Broadband Systems [44]. Therefore, traffic on both the uplink and downlink channel is grouped into (fixed length) frames. The fact that battery reductions can be achieved by using a frame structure will become apparent in the sequel of this section.

The uplink and downlink frames are synchronized in time, i.e., the header of a downlink frame is immediately followed by the start of an uplink frame (after a negligible round trip time that is captured within the guard times, see Figure 6.2). Each uplink frame consists of a (variable or fixed length) contentionless and a (variable or fixed length) contention period, where the length of the contentionless period dominates that of the contention period. An MS is allowed to transmit in the contentionless period after receiving a permit from the BS. The BS distributes the permits among the MSs based on the current requirements of each MS. Therefore, MSs must inform the BS about their current bandwidth needs using requests. Whenever an MS forwards a packet to the BS a request is piggybacked to the packet. When a packet that is generated in an MS finds the transmission queue empty (in that MS), it uses the contention period to inform the BS about its presence (i.e., it uses the contention period to sent a request). Piggybacking is not possible in such case. Notice, piggybacking is only a performance optimization and not a requirement.

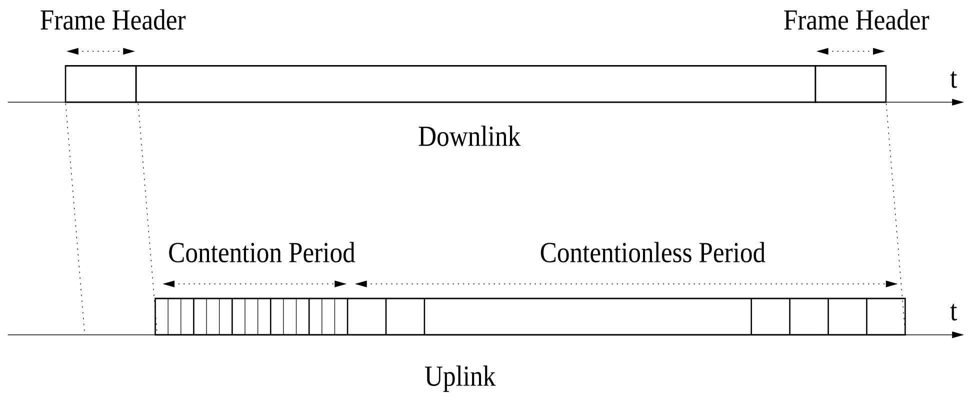


Figure 6.2: *Frame Structure*

Each downlink frame starts with a frame header in which the required feedback on the contention period of the previous uplink frame is given. This informs the MSs participating in the contention period whether a collision occurred or whether their request has been successfully received. Apart from the feedback information, the frame header contains permits for the contentionless period of the uplink frame and announces the identity of the MSs receiving a packet in the downlink frame. MSs whose identity is not mentioned can switch to the sleep mode until the start of the next downlink frame (unless they transmit something in the uplink direction).

We mentioned that piggybacking is merely an optimization. As far as the contention channel is concerned, one could also replace it by a (periodic) polling scheme. For instance, in a Passive Optical Network (PON) a periodic polling scheme is used by the Optical Network Units (ONUs) to inform the Optical Line Termination point (OLT) about their bandwidth requirements [53]. In general, a polling scheme is easy to implement, especially if the number of users remains fixed, e.g., in a wired network, but requires a reasonable amount of bandwidth (a few Mbit/s). These few Mbits are less expensive in a wired network, where one has hundreds of Mbits available, but become very expensive in a wireless network. Therefore, it might be better to use a contention channel (with piggybacking) or to combine both methods. In the next section, we present the Identifier Splitting Algorithm (ISA).

6.2 The Identifier Splitting Algorithm (ISA)

The Identifier Splitting Algorithm is characterized by two parameters:

- L : the maximum number of contention slots allowed in a single uplink frame (see Section 6.1),
- Q : the splitting factor.

The functionality of these parameters becomes apparent in the remainder of this section. Let us first introduce the notion of a contention cycle (CC). A contention cycle (CC) consists of a number of consecutive upstream frames during which the contention of all requests, present in the MSs at the beginning of the cycle, is resolved. The system is gated, in the sense that any request generated by an MS that wants to access the contention channel during a CC is blocked until the start of the next CC.

A single contention slot is available in the first frame of a CC. We refer to this slot as level 0 of the contention tree. Any MS having a request ready at the start of the CC makes use of this slot. Next, the BS checks whether a successful transmission occurred in this slot and informs the MS(s) that were involved in the scheme accordingly in the next downstream frame using a feedback field. Three situations are possible:

- The slot was empty, i.e., none of the MSs accessed the contention channel. As a result a new CC starts in the next frame.
- An MS sending its request in this slot succeeded. In this case the MS returns to the piggybacked state. Again, a new CC starts in the next frame.
- A collision occurred. In this case, the next level (level 1) of the CC provides Q contention slots. Based on the first digit of their MAC addresses, as opposed to the classical coin flip, the MSs involved split up into Q distinct sets. An MS belonging to the first set uses the first slot of level 1 to attempt a retransmission, the second set uses the second slot of level 1 and so on.

The process of generating Q slots in a level for each slot of the previous level in which a collision occurred, is repeated level after level, each time using the next digit of the Q-ary MAC address in case of a collision. Thus, during the i-th level of a CC two MSs can only collide if the first i digits of their MAC addresses are identical. Therefore, provided that the address that uniquely identifies an MS is n digits long, collisions are always resolved at level n. Notice, the number of contention slots, for each level, equals Q times the number of collisions during the previous level. Figure 6.3 shows an example of a CC with 6 participants for $Q = 2$. In this figure **CO** refers to a collision, **SU** to a success and **EM** to an empty slot. The MAC addresses of the successful MSs are added to the corresponding slot.

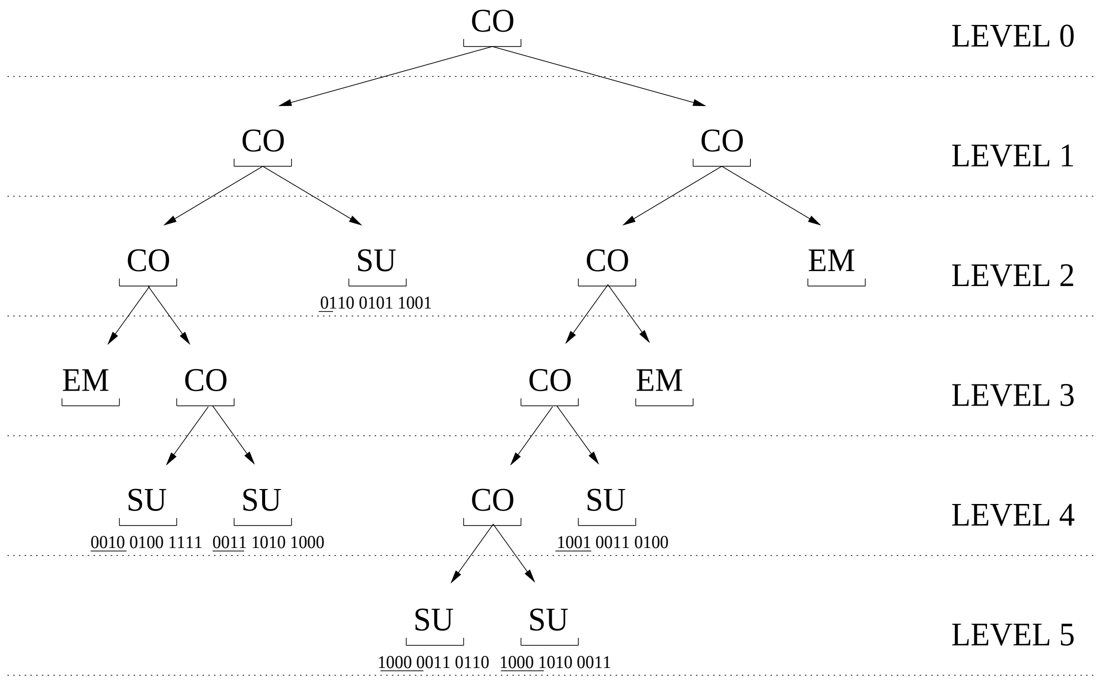


Figure 6.3: *Demonstrating ISA*

A level of the contention tree corresponds to a single frame, except when the number of slots at level i is larger than some predefined value L. This parameter L defines the maximum number of contention slots that we allow in a single frame. Thus, if a certain level of the tree requires $x = mL + j$ slots with $m \geq 0$ and $1 \leq j \leq L$ then $m + 1$ frames are required to support this level.

6.3 The Identifier Splitting Algorithm Combined with Polling

The Identifier Splitting Algorithm combined with Polling is characterized by three parameters:

- L : the maximum number of contention slots allowed in a single uplink frame (see

Section 6.1),

- Q : the splitting factor,
- N_p : the trigger value for the polling feature.

Thus, the parameter N_p is added to the scheme. One of the attractive features of the Identifier Splitting Algorithm (ISA) is that as a CC is being resolved, the BS obtains more and more information about the addresses of the MSs which are still competing. For example, if the BS notices that the tree at level i (see Figure 6.3) contains k collisions and the MAC-addresses are n digits long, then the BS concludes that the remaining competing MSs can only have kQ^{n-i} possible addresses. This follows from the fact that each slot at level i corresponds to Q^{n-i} addresses. In such case, we state that the remaining size of the MAC address space is equal to kQ^{n-i}. This information can be used by the BS in an attempt to improve the performance characteristics.

We propose the following method: when the size of the remaining MAC address space Y becomes smaller than some predefined value, say N_p, the protocol switches to polling. Polling, in this context, means that one slot is provided for each address in the remaining address space. Depending on the relationship between L and $Y(\leq N_p)$, one or multiple frames are required to support polling. The introduction of the parameter N_p, referred to as the trigger value, not only allows us to improve the performance of the ISA scheme, but also provides some additional challenges as far as the performance evaluation is concerned because it creates additional dependencies between a number of random variables (see Chapter 7).

6.4 Skipping the First Few Levels

In the previous two sections the contention period of the first frame of a CC consisted of a single contention slot (level 0 of the contention tree). Now we drop this condition: instead of starting with just one contention slot in the first frame, we provide more than one slot during the first frame of a CC. The idea to offer more than 1 slot for the first transmission attempt is far from uncommon in splitting algorithms with blocked access [51, 63] [3, p291]. The starting level is said to be S_l, with $0 \leq S_l \leq n$, if the first frame of the CC contains Q^{S_l} contention slots. An MS taking part in the contention cycle selects one of these Q^{S_l} slots based on the first S_l digits of its n-digit MAC address. We need $\lceil Q^{S_l}/L \rceil$ frames to support the starting level S_l.

The starting level S_l can either be fixed at a predefined value or can change in time. A fixed starting level S_l is expected to have a positive impact on the delay. Apart from that, the throughput might improve in case of high loads [51]. Unfortunately, as shown in the numerical results, this results in some additional throughput losses during low load periods. To solve this we propose a scheme that changes the starting level dynamically between level S_{min} and S_{max}. To make this decision the system load ρ is not taken into account, as this value is hard to measure or predict in real systems. We therefore use the length of the previous CC as follows.

The starting levels are defined using the following two threshold values: B_l and B_m. Suppose that at some point in time the starting level of a CC equals S_l and let L be the length of this CC, then the new starting level S_l' obeys the following equation:

$$S_l' = \begin{cases} \max(S_l - 1, S_{min}) & L \leq B_l \\ S_l & B_l < L < B_m \\ \min(S_l + 1, S_{max}) & L \geq B_m \end{cases} . \tag{6.1}$$

Clearly all MSs wanting to access the contention channel need to be aware of the current starting level. We suggest that this knowledge is broadcasted by the BS at the start of every CC. Therefore, it is not necessary for all MSs, including those that do not use the contention channel, to keep track of the lengths of the CCs.

6.5 Multiple Instances of ISA

In what follows we demonstrate by means of an example how multiple instances of the ISA protocol can be created from a single instance with a fixed starting level $S_l \geq 1$. For this example (see Figure 6.4) the starting level S_l is fixed at 1 and Q is set at 2. As can be seen in Figure 6.4 all collisions in the right-hand side of the tree are resolved at level 3. Suppose that during these 3 levels a number of MSs, not necessarily participating in

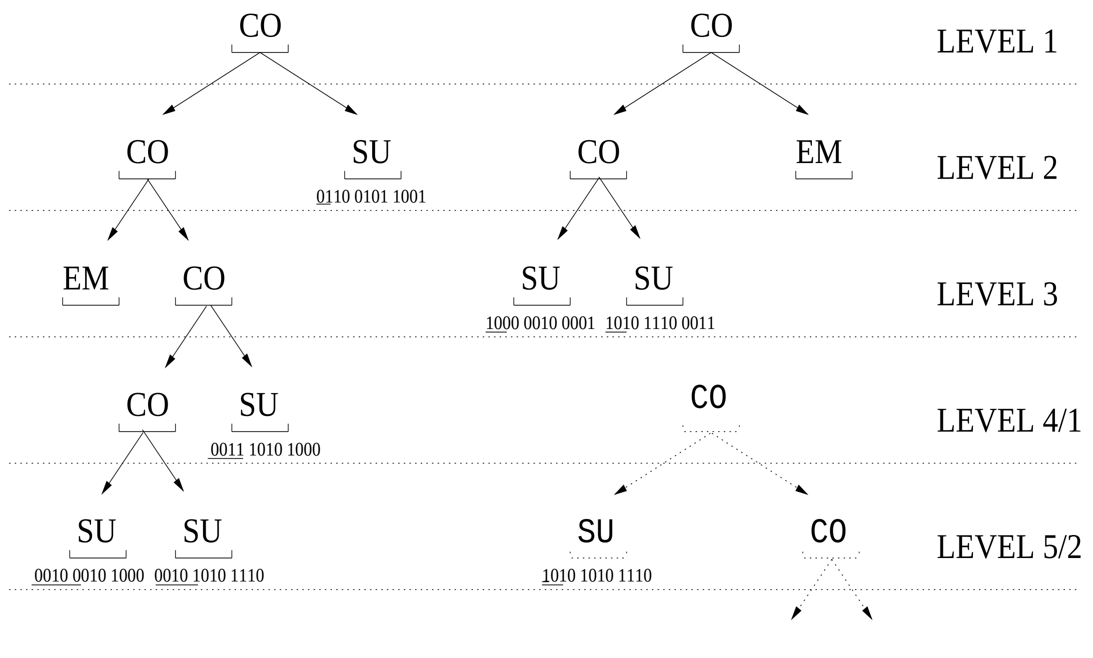


Figure 6.4: *Creating Multiple Instances of ISA*

this CC and some with the first bit of their MAC address equal to 1, have generated a new request. Then, according to the previous sections, they have to wait until the start of the next CC; that is, until the end of level 5. Alternatively, the BS may initiate a new instance of the ISA protocol, to be used by all MSs belonging to the second half of the tree, thereby creating a second instance of the ISA protocol. The first instance is used by all MSs whose first bit of their address equals 0, the second half is devoted to the other MSs, the MAC addresses of which start with a 1.

In general, the ISA protocol with starting level S_l and a splitting factor Q can be uncoupled to form Q^{S_l} different instances of ISA, where each instance corresponds with a partition of the address space. Another advantage of this method is that the contention slots are spread more uniformly over consecutive frames, as the different instances are not necessarily in phase, i.e., the tops of the different trees might occur in different frames. The disadvantage of uncoupling is that we can no longer decrease the starting level below level S_l. It is possible to combine multiple instances and polling, but we do not consider it.

6.6 MS Behaviour

The behaviour of the different MSs, using a single instance of the ISAP scheme, is described in Figure 6.5 by means of a flow chart. The following notations are used. An arrow that is accompanied by a capital S indicates that the transition is made at the end of a frame header, i.e., after receiving the feedback information from the BS. Let A be a MAC address, A_i the i-th digit of the MAC address A, $v_f(A,i)$ the integer value denoted by the first i digits of the MAC address A and $v_l(A,i)$ the integer value denoted by the last i digits of the MAC address A. For instance for $Q = 2$, $v_f(101101, 3) = 5$ and $v_l(101101, 4) = 13$.

As long as an MS is not using the contention channel, it remains in the *inactive* state. An MS that generates a request makes a transition to the *blocked* state. There it remains until the current CC is solved, checking the feedback field at the beginning of every frame. The feedback field, which is present in the downlink frame header, contains the following subfields: a bit, denoted as the l-bit, which indicates whether the current level has finished, a bit, denoted as the c-bit, which indicates whether the current CC has finished (if set: the starting level of the next CC is also included), an integer value Tc denoting the number of collisions that occurred (so far) at the current level and a set of bits, one for each contention slot, where a 0 indicates a success (or an empty slot) and a 1 a failure. Notice that one feedback bit for each contention slot is sufficient as we do not take capture effects into account.

Once the current CC has ended—that is, the c-bit is set—three parameters 'Lvl', 'Pos' and 'Offset' are initialized. They have the following function:

- Lvl : indicates the current level of the CC, therefore its value is incremented by one at the start of each level during a CC.
- Pos : is a variable that holds the number of the contention slot to be used by the MS (the slots are numbered starting from 1).
- Offset : an integer value that keeps track of the number of slots, belonging to the current level, present in prior frames.

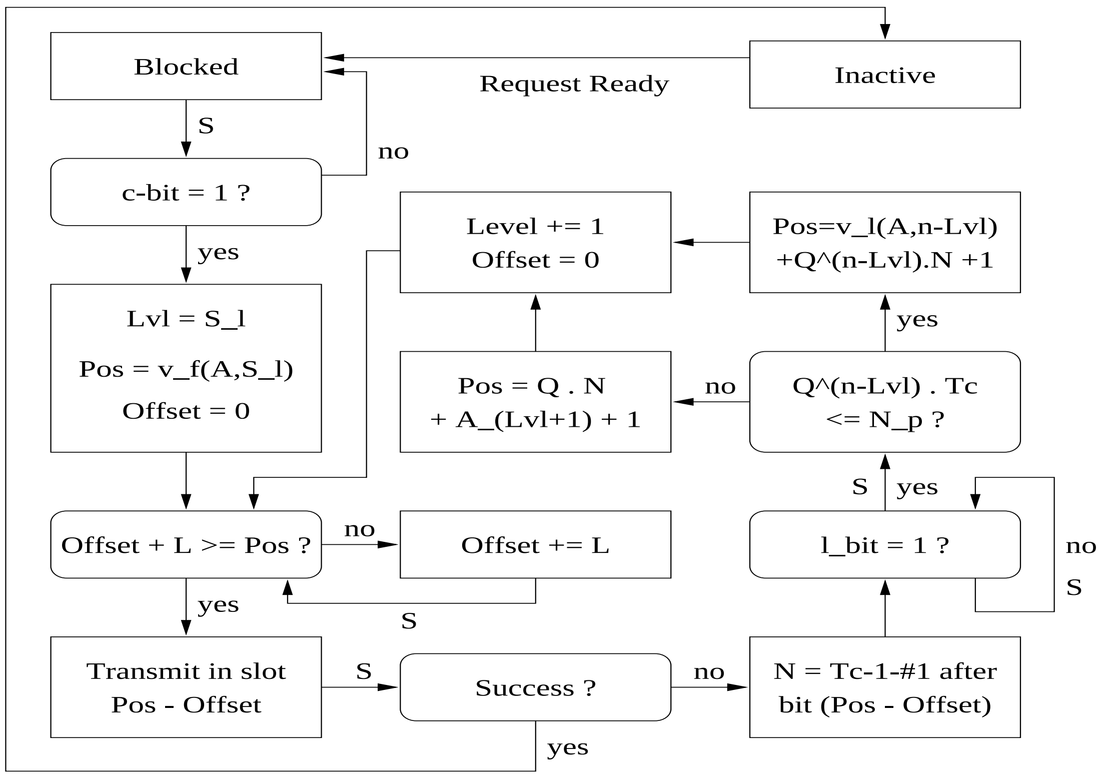


Figure 6.5: *The Flow Chart of an MS (A = MAC address of the MS)*

After initialization the MS waits for the frame that contains slot number 'Pos' by making use of the 'Offset' value. Next, the *transmission* state is entered. A transmission takes place in slot number 'Pos' and the result is found by checking the corresponding feedback bit. If successful we return to the *inactive* state, otherwise the MS sets the parameter N and waits until the current level has finished. N indicates the number of collisions that have occurred at the current level before slot number 'Pos'. Next, the MS checks to see whether a switch to polling is made and depending on this result assigns a new value to 'Pos'. This routine is repeated until a successful transmission occurs. Checking to see whether a switch to polling is made at level $i+1$ simply consists of calculating the size of the remaining address space and comparing the result with N_p. Due to the fact that a slot at level i corresponds with Q^{n-i} addresses, the size of the remaining address space is found by multiplying the number of collisions at that level i by Q^{n-i}.

6.7 More Optional Parameters

The numerical examples presented in Chapter 8 indicate that the polling feature has a positive impact on the delay. It does however reduce the throughput achieved on the contention channel. In order to limit the throughput losses caused by the polling feature, one could add another parameter M_p to the ISAP scheme. Whenever the ISAP scheme is enriched by the M_p parameter, we refer to it as the M-ISAP scheme.

The aim of M-ISAP is to guarantee a minimum throughput on the polling slots. Notice, ISAP already guarantees a worst case throughput T_{poll} on the polling slots of

$$T_{poll} = \frac{2}{Q^{\lfloor \log_Q N_p \rfloor}}. \tag{6.2}$$

For instance, for $Q = 3$ and $N_p = 30$, we get $T_{poll} = 2/27$. Equation (6.2) can be understood as follows. The worst possible throughput on the polling slots is reached when the switch to polling occurs as a result of a single collision, containing two contenders, at the highest possible level. Looking at equation (6.2), ISAP hardly provides any guarantee for large values of N_p. A better guarantee is achieved by prohibiting ISAP to switch to polling until a certain level, say M_p, is reached. Indeed, the slots used for polling have a minimum throughput of T_{poll} of

$$T_{poll} = \frac{2}{\min(Q^{n-M_p+1}, Q^{\lfloor \log_Q N_p \rfloor})}, \tag{6.3}$$

where n is the length of a MAC address. For instance, for $Q = 2, n = 8, N_p = 35$ and $M_p = 6$ we get a worst case throughput of 0.25 on the polling slots. In the same scenario ISAP would only guarantee a throughput of 0.0625 on the polling slots.

Chapter 7

Analysis of the Identifier Splitting Algorithm combined with Polling

In this chapter we study the ISAP algorithm by means of several analytical models [65–67]. The main objective of these models is to obtain experience that allows a well-founded understanding of the impact of the different protocol parameters and to reveal possible delay vs. throughput tradeoffs. Numerical results of this study are presented in Chapter 8. Petras, *et al* [50–52] have studied the first two moments of the length of an ISA CC with k contenders, by means of recursive formulas. Their main assumption is that every level can be supported by a single frame, i.e., the parameter L is not taken into account. Fernandez and Sallent [15] have studied none deterministic splitting algorithms in a hybrid Fiber-Coax Broadband Access Network by means of functional equations. They also traverse the contention tree in a breadth-first approach. The access network considered by Fernandez does not contain a frame structure: the system is slotted and each level of the contention tree is separated by B contentionless time slots. Therefore, the analysis is very different from ours.

This chapter is subdivided into five sections. A number of simplifying assumptions are made in Section 7.1, that apply to all the analytical models presented. In Section 7.2, we start by introducing a model for the binary ISA scheme. We continue with the binary ISAP scheme, i.e., we add the polling feature. Next, we consider fixed and variable starting levels and multiple instances. In Section 7.3, we generalize these models to the Q-ary case. Section 7.4, indicates how to evaluate M-ISAP (see Section 6.7). In Sections 7.2 to 7.4 the parameter L is not taken into account (see Section 7.1). Finally, in Section 7.5, we calculate some important expected values that provide insight on the interaction of the parameter L with the other protocol parameters.

7.1 Assumptions

Let n be the size of the MAC-addresses (in digits). The number of MSs located within the reach of the BS is assumed to be Q^n—that is, all MAC addresses are utilized. Furthermore, the aggregate traffic, generated by all MSs on the uplink contention channel, is assumed to have a Poisson distribution with a mean of λ requests per frame. As the number of MSs is finite and equals Q^n, the number of requests generated during a CC should never exceed Q^n. Therefore, we drop at random some of the arrivals if this value is exceeded (for $x > Q^n$ arrivals, we drop $x - Q^n$ arrivals). Alternatively, we could drop the last $x - Q^n$ arrivals. The fact that we drop these arrivals at random (instead of dropping the last $x - Q^n$ arrivals) should hardly have any influence on the numerical examples presented, because the probability of having more than Q^n arrivals during a CC is negligible. Random dropping assures that the requests arrive in a uniform way during a CC. Hence, define the random variable I_i as the number of requests generated during a CC consisting of i frames, then

$$P[I_i = k] = \frac{(\lambda i)^k}{k!} e^{-\lambda i}, k < Q^n \tag{7.1}$$

$$P[I_i = Q^n] = \sum_{k \geq Q^n} \frac{(\lambda i)^k}{k!} e^{-\lambda i}. \tag{7.2}$$

Notice, we do not need to consider bursty input traffic since we are observing the access channel used by an MS that wants to transmit a request after a period of silence. In real-life systems the following holds with respect to the number of MSs participating and their addresses:

- MSs that were successful during the last frame of a CC will never participate in the next CC.
- Participating MSs, regardless of the frame in which they were successful, are less likely to take part in the next CC as opposed to those that did not participate at all.

To keep the model analytically tractable, both these remarks are ignored. Thus, the addresses of the MSs taking part in the scheme at the beginning of a CC are uniformly distributed over the complete address space and their number is distributed according to a Poisson distribution, where the mean depends on the length of the previous CC.

In the first three parts (Sections 7.2 to 7.4), we assume that each level of the CC corresponds with a single frame, i.e., L is assumed to be large enough to support any level of the splitting algorithm. Therefore, we cannot use the model to study a system in which the contention channel is highly loaded. Notice that each level of a CC can always be supported by a single frame if a CC has $k \leq 2L/Q$ participants, whatever the addresses of these k stations might be. Indeed, a level requires $x > L$ slots whenever x/Q collisions occurred during the previous level. In order to have y collision we need at least $2y$ MSs participating. In Sections 7.2 to 7.4, N_p and Q^{S_l} are smaller than or equal to L. In the fourth part (Section 7.5), we drop the assumption on L and consider all N_p and S_l values.

7.2 Analysis of the Binary ISAP Protocol

The work presented in this section was published in [66]. In the first subsection we calculate the throughput and the delay density function of the ISA scheme (Section 7.2.1). In a second subsection we focus on ISAP with a fixed starting level $S_l = 0$ (Section 7.2.2). Afterwards, we consider other fixed starting levels (Section 7.2.3) and variable starting levels (Section 7.2.4). The following random variables will be used in the sequel of this section.

- X_c, resp. X_a, denotes the number of contenders or participants in a CC for the ISA, resp. ISAP protocol.
- R_c, resp. R_a, denotes the level at which the CC is resolved (i.e., the number of frames needed minus one) for the ISA, resp. ISAP scheme.
- $C_i^{(c)}$, resp. $C_i^{(a)}$, denotes the number of collisions at level i for both protocols. These variables range from 0 to 2^i.
- P_a denotes the level at which we poll for the ISAP scheme. If the scheme is solved without polling we let P_a be equal to $n+1$.

Furthermore we use the symbol C_r^n to denote the number of different possible combinations of r from n different items.

7.2.1 The Identifier Splitting Algorithm (ISA)

The Delay Analysis

(A) We start by studying the random variable R_c conditioned on X_c. Notice that at level i the address space is split into 2^i equal parts of size 2^{n-i}. For the scheme to be collision free at level i we can only allow one participating MS in each subspace. This results in

$$P[R_c \leq i \mid X_c = k] = \frac{2^{(n-i)k} C_k^{2^i}}{C_k^{2^n}}. \tag{7.3}$$

This can be proven by noticing that $P[R_c \leq i \mid X_c = k] = P[R_c \leq i \mid X_c = k - 1]\, 2^{n-i}\, (2^i - (k-1))/(2^n - (k-1))$ using induction on k. An alternative proof is based on the multivariate hypergeometric distribution. By subtraction we obtain $P[R_c = i \mid X_c = k]$, which is denoted as $p_c(k, i+1)$ (we write $i+1$ to indicate the number of frames used).

(B) Let us now focus on X_c. Clearly X_c is the steady-state vector of the Markovian process $(X_n^{(c)})_n$, where $X_n^{(c)}$ denotes the number of contenders during the n-th CC. Due to **(A)**,

$$t_c(k,j) \overset{\text{def}}{=} P[X_{n+1}^{(c)} = j \mid X_n^{(c)} = k] = \sum_{t=1}^{n+1} \frac{(\lambda t)^j e^{-\lambda t}}{j!} p_c(k,t), \tag{7.4}$$

for $0 \le j \le 2^n - 1$. For $j = 2^n$ we assign the remaining probability mass. X_c is then found by solving the eigenvector problem. Applying the definition of the expected value gives us the mean number of participants $E[X_c]$ in a CC.

(C) Before we can calculate the delay we still need to make the following observation. Consider an arbitrary arrival in a CC, then we need to know the probability that this CC is k frames long and that there will be l contenders in the next CC. We denote $L_{cur}^{(c)}$ and $X_{next}^{(c)}$ as the length of the CC in which an arbitrary arrival occurs and the number of participants in the next CC. Some straightforward reasoning shows that the following relationships between $X_{next}^{(c)}$, $L_{cur}^{(c)}$ and X_c hold:

$$P[X_{next}^{(c)} = l] = \frac{P[X_c = l]l}{E[X_c]} \tag{7.5}$$

and

$$P[L_{cur}^{(c)} = l] = \lambda l \sum_{k=0}^{2^n} P[X_c = k] p_c(k,l)/E[X_c] \tag{7.6}$$

where $p_c(k,l)$ was defined in **(A)**. Notice, $\sum_l P[L_{cur}^{(c)} = l](\lambda l)^{k-1}/(k-1)!\, e^{-\lambda l} = P[X_{next}^{(c)} = k]$. Let us prove equality (7.5) (equation (7.6) can be found in a similar way). Consider a finite set of N of consecutive CCs. Denote $T_i(N)$ as the number of CCs during which i arrivals occur. Then, the probability that an arbitrary arrival competes, during a CC, with $i-1$ other arrivals equals $iT_i(N)/(\sum_i iT_i(N))$. As $N \to \infty$, $T_i(N)/N$ approaches $P[X_c = i]$ and $\sum_i iT_i(N)/N$ approaches $E[X_c]$.

Combining (A), (B) and (C) Having the results from **(A)**, **(B)** and **(C)**, we can calculate the mean delay. Clearly the delay consists of two parts. The first part D_1 is the time until the start of the next CC and the second part D_2 is the number of frames needed until our tagged request is successful. Using expression (7.6) and knowing that the arrivals are distributed uniformly within a CC (see Section 7.1), the expected value for the first part equals

$$E[D_1] = \sum_{i=1}^{n+1} P[L_{cur}^{(c)} = i] i/2. \tag{7.7}$$

By definition of the expected value the second part equals

$$E[D_2] = \sum_{i=0}^{n} \sum_{k \ge 1} P[X_{next}^{(c)} = k](i+1)(\mathcal{F}_c(i,k) - \mathcal{F}_c(i-1,k)), \tag{7.8}$$

where $\mathcal{F}_c(i,k)$ denotes the probability that a tagged request is successful at or before level i given that there where $k-1$ other contenders ($\mathcal{F}_c(-1,k)$ is zero in the expression above). Again we can prove by induction that

$$\mathcal{F}_c(i,k) = \frac{C_{k-1}^{2^n-2^{n-i}}}{C_{k-1}^{2^n-1}}. \tag{7.9}$$

Adding $E[D_1]$ and $E[D_2]$ results in the mean delay.

The delay density function Using **(A), (B)** and **(C)**, it is easy to show that the delay density function $D_c(x)$ (with x between 1 and $2(n+1)$) is given by the following step function:

$$D_c(x) = \sum_{s=1}^{\lfloor x \rfloor} \sum_{j=\lceil x \rceil - s}^{n+1} \sum_{l=1}^{2^n} \frac{\mathcal{F}_c(s-1,l) - \mathcal{F}_c(s-2,l)}{j} \mathcal{G}_j(l) P[L_{cur}^{(c)} = j], \tag{7.10}$$

where $\mathcal{G}_j(l) = \frac{(\lambda j)^{l-1}}{(l-1)!} e^{-\lambda j}$ for $l < 2^n - 1$ and $\mathcal{G}_j(2^n) = \sum_{j \geq 2^n-1} \frac{(\lambda j)^{l-1}}{(l-1)!} e^{-\lambda j}$. In (7.10) s denotes the number of transmissions (including the successful transmission) a tagged request needs, j refers to the length (in frames) of the CC in which our tagged request is generated and $l-1$ equals the number of other competitors apart from our tagged one.

The Throughput Analysis

In this section we determine the throughput of the ISA scheme. First, define two more sets of random variables $S_i^{(c)}$ and $S_i^{(a)}$, being the number of slots used at level i by both schemes. From the foregoing we already obtained $P[X_c = k]$; thus, the throughput T_c is found as

$$T_c = \frac{E[X_c]}{\sum_{k=0}^{2^n} P[X_c = k] E[\sum_i S_i^{(c)} \mid X_c = k]}. \tag{7.11}$$

We could calculate the expected number of slots in this formula as was done in [51] (using a strong recursive scheme). Still, it is possible to get the same results using a more direct approach as follows. First, notice that

$$E[\sum_i S_i^{(c)} \mid X_c = k] = 1 + \sum_{i=1}^{n} E[S_i^{(c)} \mid X_c = k]. \tag{7.12}$$

On the other hand we know that the expected number of slots at level i equals twice the expected number of collisions at level $i-1$, while the expected number of collisions at level i matches

$$E[C_i^{(c)} \mid X_c = k] = 2^i \left(1 - \frac{C_k^{2^n-2^{n-i}}}{C_k^{2^n}} - 2^{n-i} \frac{C_{k-1}^{2^n-2^{n-i}}}{C_k^{2^n}} \right). \tag{7.13}$$

Hence, by substituting the summation index

$$E[\sum_i S_i^{(c)} \mid X_c = k] = 1 + 2^{n+1} \sum_{i=1}^{n} \left[2^{-i} \left(1 - \frac{C_k^{2^n - 2^i}}{C_k^{2^n}} \right) - \frac{C_{k-1}^{2^n - 2^i}}{C_k^{2^n}} \right]. \tag{7.14}$$

As it turns out, the right-hand side of equation (7.14) was already obtained by Trabb Pardo [17] in 1977. At that time splitting algorithms were not yet invented, but these quantities are also relevant to the analysis of tries in computer algorithms.

7.2.2 The Identifier Splitting Algorithm combined with Polling

The Delay Analysis

In this section we will follow the same lines of reasoning as in Section 7.2.1 and we start by studying $P[R_a \leq i \mid X_a = k]$.

(A') Two cases can be considered. First, the CC might be solved before level i or at level i due to polling, secondly, it might be solved at level i without a switch to polling. Hence,

$$\begin{aligned} P[R_a \leq i \mid X_a = k] = P[R_a \leq i - 1 \cup P_a \leq i \mid X_a = k] + \\ P[R_a = i \cap P_a > i \mid X_a = k]. \end{aligned} \tag{7.15}$$

The first probability is discussed in **(A1')**, the second in **(A2')**.

(A1') We calculate the complementary probability mass. By definition of the polling mechanism (see Section 6.3) we have

$$P[R_a \geq i \cap P_a > i \mid X_a = k] = P[C_{i-1}^{(a)} > \left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor \mid X_a = k]. \tag{7.16}$$

The right-hand side is found using the following relationship:

$$P[C_{i-1}^{(a)} = \left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor + x \mid X_a = k] = P[C_{i-1}^{(c)} = \left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor + x \mid X_c = k], \tag{7.17}$$

for $x \geq 1$, but not necessarily for $x \leq 0$. To prove this we must show that having $\left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor + x$ collisions at level $i - 1$ given k contenders for ISA implies the same for ISAP and vice versa. Clearly, if ISAP had this number of collisions (at level $i - 1$ given k participants), polling did not occur before; thus, we have the same effect for ISA. On the other hand if the ISA scheme results in that many collisions, ISAP could not have switched to polling because the remaining address space is too large at level $i - 1$ and can only decrease in size as the level increases.

Our main objective now is to find the probability that we have exactly l collisions at level i given k participants (when using the ISA scheme). Although these values are easy to describe mathematically by a sum of multivariate hypergeometic probabilities, this is of no practical use due to the high computational complexity. A more complicated but appropriate way would be to apply the Inclusion-Exclusion Principle [29, 73]. Due to the alternating sign, this method tends to give numerical problems for large values of n. We propose the following variation on the Inclusion-Exclusion Principle (where the first equality is a consequence of (7.3)):

$$s(i, 2^i, k) = \frac{2^{(n-i)k} C_k^{2^i}}{C_k^{2^n}}, \tag{7.18}$$

$$s(i, l, k) = C_l^{2^i} \sum_{l_1=0}^{l} 2^{(n-i)l_1} \frac{C_{l_1}^{l} C_{k-l_1}^{2^n - l2^{n-i}}}{C_k^{2^n}} - \sum_{x=1}^{2^i - l} C_l^{l+x} s(i, l+x, k), \tag{7.19}$$

where $s(i,l,k) = P[C_i^{(c)} = 2^i - l \mid X_c = k]$. We can use (7.18) and (7.19) in a floating point environment for $n \leq 7$.

(A2') In this case each collision at level $i-1$ involves only two MSs, otherwise they cannot be solved at level i. The probability that such a collision is solved, at level i, equals $\frac{2^{n-i}}{2^{n-i+1}-1}$. Thus by means of the multivariate hypergeometric distribution we get

$$P[R_a = i \cap P_a > i \mid X_a = k] = \sum_{u=u_{i-1}+1}^{\lfloor \frac{k}{2} \rfloor} 2^{(n-i+1)(k-2u)} \frac{\left(C_2^{2^{n-i+1}}\right)^u C_u^{2^{i-1}} C_{k-2u}^{2^{i-1}-u}}{C_k^{2^n}} \left(\frac{2^{n-i}}{2^{n-i+1}-1}\right)^u, \tag{7.20}$$

where u_i denotes $\left\lfloor \frac{N_p}{2^{n-i}} \right\rfloor$.

(B',C') Let us define $p_a(k, i+1)$ as $P[R_a = i \mid X_a = k]$. Steps **(B')** and **(C')** are straightforward to obtain from **(B)** and **(C)**. To calculate the mean delay, we need to find $\mathcal{F}_a(i,k)$, i.e., the probability that a tagged request is successful at or before level i given that we had k contenders (for the ISAP scheme). We denote $u_{i-1}+1$ as v_i; thus, $v_i = 1 + \lfloor \frac{N_p}{2^{n-i+1}} \rfloor$. In **(A1')** it was argued that the event $C_{i-1}^{(c)} \geq v_i$ is the same as $C_{i-1}^{(a)} \geq v_i$, when conditioned on X_c, resp. X_a, which in its turn coincides with $P_a > i \cap R_a \geq i$. Hence,

$$\mathcal{F}_a(i,k) = P[R_a \leq i-1 \cup P_a \leq i \mid X_a = k] + \sum_{s \geq v_i} P[R_t \leq i \cap C_{i-1}^{(c)} = s \mid X_c = k], \tag{7.21}$$

where R_t denotes the level at which our tagged request is successful. This first probability was found in **(A1')**. The second one is found using the methodology of equations (7.18) and (7.19) as follows. We define $t(i,s,k)$ as $P[R_t \le i \cap C_{i-1}^{(c)} = 2^{i-1} - s \mid X_c = k]$. We get (where the first equation is a consequence of (7.3))

$$\begin{aligned} t(i, 2^{i-1}, k) &= \frac{2^{(n-i+1)k} C_k^{2^{i-1}}}{C_k^{2^n}} &(7.22)\\ t(i,s,k) &= C_s^{2^{i-1}} \sum_{l_1=0}^{s} 2^{(n-i+1)l_1} \frac{C_{l_1}^s C_{k-l_1}^{2^n - s2^{n-i+1}}}{C_k^{2^n}} \cdot \\ & \left(\frac{l_1}{k} + \left(1 - \frac{l_1}{k}\right) \frac{C_{k-l_1-1}^{2^n - s2^{n-i+1} - 2^{n-i}}}{C_{k-l_1-1}^{2^n - s2^{n-i+1} - 1}} \right) - \sum_{x=1}^{2^{i-1}-s} C_s^{s+x} t(i, s+x, k). &(7.23) \end{aligned}$$

When we look at the delay density function we can make use of formula (7.10) (where the indices a are used instead of c). This concludes the delay analysis.

The Throughput Analysis

Since we already know the probabilities $P[X_a = k]$ from the delay analysis, it is sufficient to find $E[\sum_i S_i^{(a)} \mid X_a = k]$. Unfortunately this is not as straightforward as one might expect. We start in a similar manner as in the previous section. The expected number of slots used equals the sum of the expected number of slots used at each level. By definition of the ISAP scheme we have

$$\begin{aligned} E[S_i^{(a)} \mid X_a = k] = P[P_a = i \mid X_a = k]\ E[S_i^{(a)} \mid X_a = k \cap P_a = i] + \dots \\ P[P_a > i \cap R_a \ge i \mid X_a = k]\ E[S_i^{(a)} \mid X_a = k \cap P_a > i \cap R_a \ge i], \quad (7.24) \end{aligned}$$

by observing that the expected number of slots is zero if $R_a \le i-1$. The second probability was obtained in **(A1')**, the first one is calculated as $P[R_a \le i-1 \cup P_a \le i \mid X_a = k]$ minus $P[R_a \le i-1 \mid X_a = k]$, two results that were also obtained in **(A')**. The computation of both expected values remains (for $i \ge 2$ since $S_0^{(a)}$ and $S_1^{(a)}$ are trivial to obtain). They are discussed in **(D')** and **(E')**.

(D') First consider $E[S_i^{(a)} \mid X_a = k \cap P_a > i \cap R_a \ge i]$. In this case the number of slots used at level i equals two times the number of collisions at level $i-1$. Also in **(A1')** it was shown that the event $P_a > i \cap R_a \ge i$ is the same as $C_{i-1}^{(c)} \ge v_i$, when conditioned on X_a, resp. X_c. Thus it is sufficient to find

$$E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-1}^{(c)} \ge v_i].$$

This expected value is obtained using the definition of the expected value combined with (7.17)

$$E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-1}^{(c)} \ge v_i] = v_i + \sum_{s=0}^{2^{i-1}-v_i} s \frac{P[C_{i-1}^{(c)} = s + v_i \mid X_c = k]}{P[C_{i-1}^{(c)} \ge v_i \mid X_c = k]}, \quad (7.25)$$

where we applied the following proposition. If an event $A \subseteq C$ then $P[A \mid B \cap C]$ equals $P[A \mid B]/P[C \mid B]$. In this case, A is equal to $C_{i-1}^{(c)} = s + v_i$ and C is chosen as $C_{i-1}^{(c)} \geq v_i$.

(E') As opposed to the first case, i.e., **(D')**, the expected number of slots equals 2^{n-i+1} times the expected number of collisions at level $i-1$ provided that we do poll at level i and we have k contenders. Also, since the event $P_a = i$ is the same as $R_a \geq i \cap C_{i-1}^{(a)} < v_i$ we are actually looking for

$$E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i \cap C_{i-1}^{(a)} < v_i]. \tag{7.26}$$

We start with the following observation:

$$\begin{aligned} &E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i] = \\ &\quad P[C_{i-1}^{(a)} \geq v_i \mid X_a = k \cap R_a \geq i]\ E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i \cap C_{i-1}^{(a)} \geq v_i] + \ldots \\ &\quad P[C_{i-1}^{(a)} < v_i \mid X_a = k \cap R_a \geq i]\ E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i \cap C_{i-1}^{(a)} < v_i], \end{aligned} \tag{7.27}$$

where the expression of interest is part of the right-hand side. Both probabilities are clearly each others' complement; thus, it is sufficient to calculate the first. To do this remark again that if an event $A \subseteq C$ then $P[A \mid B \cap C]$ equals $P[A \mid B]/P[C \mid B]$. Applying this result with A equal to $C_{i-1}^{(a)} \geq v_i$ and with C as $R_a \geq i$, (A is a part of C because $v_i > 0$) yields the following expression for the first probability:

$$P[C_{i-1}^{(a)} \geq v_i \mid X_a = k]/P[R_a \geq i \mid X_a = k]. \tag{7.28}$$

Both these values were obtained in section **(A')**. Again two expected values remain unknown, **(E1')** and **(E2')** are devoted to them.

(E1') We start with the one in the right-hand side. Notice that event $C_{i-1}^{(a)} \geq v_i$ is a part of the event $R_a \geq i$ (as mentioned above) and this first event is the same as $C_{i-1}^{(c)} \geq v_i$ when conditioned on X_a and X_c respectively. Thus the expression we are looking for is reduced to (7.25).

(E2') Remark that the event $R_a \geq i$ coincides with $C_{i-1}^{(a)} > 0$. As the event $C_{i-2}^{(a)} \geq v_{i-1}$ contains this last event, we can also write it as $C_{i-2}^{(a)} \geq v_{i-1} \cap C_{i-1}^{(a)} > 0$. So, we want to find

$$E[C_{i-1}^{(a)} \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1} \cap C_{i-1}^{(a)} > 0]. \tag{7.29}$$

Some straightforward reasoning based on the definition of the expected value yields

$$\begin{aligned} &E[C_{i-1}^{(a)} \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1} \cap C_{i-1}^{(a)} > 0] = E[C_{i-1}^{(a)} \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1}]/ \\ &\quad (1 - P[C_{i-1}^{(a)} = 0 \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1}]). \end{aligned} \tag{7.30}$$

Applying $P[A \mid B \cap C] = P[A \cap C \mid B]/P[C \mid B]$ we find the probability in the denominator

$$P[C_{i-1}^{(a)} = 0 \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1}] = \frac{P[C_{i-1}^{(a)} = 0 \cap C_{i-2}^{(a)} \geq v_{i-1} \mid X_a = k]}{P[C_{i-2}^{(a)} \geq v_{i-1} \mid X_a = k]}, \tag{7.31}$$

due to our discussion in **(A1')** we can substitute the super- and subscripts a for c in both probabilities without altering their values. Having done this we use (7.18) and (7.19) for the computation of the denominator, while the numerator is obtained based on a similar argument as in **(A2')**

$$P[C_{i-1}^{(c)} = 0 \cap C_{i-2}^{(c)} \geq v_{i-1} \mid X_c = k] =$$
$$\sum_{u=v_{i-1}}^{\lfloor \frac{k}{2} \rfloor} 2^{(n-i+2)(k-2u)} \frac{\left(C_2^{2^{n-i+2}}\right)^u C_u^{2^{i-2}} C_{k-2u}^{2^{i-2}-u}}{C_k^{2^n}} \left(\frac{2^{n-i+1}}{2^{n-i+2}-1}\right)^u. \tag{7.32}$$

The expression is the same as in **(A2')**, but with $i-1$ substituted for $i-2$ (remember that $v_i = 1 + u_{i-1}$).

We end with the determination of the expected value in the right-hand side of (7.30). Again, we can substitute the sub- and superscripts a for c. Then, using the definition of the expected value we get

$$E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-2}^{(c)} \geq v_{i-1}] =$$
$$\sum_{l \geq v_{i-1}} E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-2}^{(c)} = l] \frac{P[C_{i-2}^{(c)} = l \mid X_c = k]}{P[C_{i-2}^{(c)} \geq v_{i-1} \mid X_c = k]}. \tag{7.33}$$

Finally, we calculate the numerator of this sum using the same methodology as in (7.18) and (7.19), where we define $e(i-1, s, k)$ as $E[2^{i-1} - C_{i-1}^{(c)} \mid X_c = k \cap C_{i-2}^{(c)} = 2^{i-2} - s]P[C_{i-2}^{(c)} = 2^{i-2} - s \mid X_c = k]$. This results in the following equations (the first equation is a consequence of (7.3)):

$$e(i-1, 2^{i-2}, k) = 2^{i-1} \frac{2^{(n-i+2)k} C_k^{2^{i-2}}}{C_k^{2^n}} \tag{7.34}$$

$$e(i-1, s, k) = C_s^{2^{i-2}} \sum_{l_1=0}^{s} 2^{(n-i+2)l_1} \left(2s + (2^{i-1} - 2s) \frac{C_{k-l_1}^{m_i} + 2^{n-i+1} C_{k-l_1-1}^{m_i}}{C_{k-l_1}^{2^n - s2^{n-i+2}}}\right)$$
$$\cdot \frac{C_{l_1}^s C_{k-l_1}^{2^n - s2^{n-i+2}}}{C_k^{2^n}} - \sum_{x=1}^{2^{i-2}-s} C_s^{s+x} e(i-1, s+x, k) \tag{7.35}$$

with m_i equal to $2^n - s2^{n-i+2} - 2^{n-i+1}$.
Remark that the expected number of slots used in this scheme given that we had k contenders is independent of the way the slots (polling and contention slots) are incorporated into the frame structure. Only the probability of having k contenders depends upon the frame structure.

7.2.3 Skipping the First Few Levels (STATIC)

In this section we describe the necessary adaptations to Section 7.2.2 in order to evaluate the ISAP scheme when some of the first levels of the tree are skipped. The performance of ISA with a higher starting level is obtained by setting N_p equal to 0. The starting level is denoted by S_l. The following random variables are defined:

- X_a^+ : the number of participants in a CC, this variable ranges from 0 to 2^n.
- R_a^+ : the level at which the ISAP scheme is resolved, this variable ranges from S_l to n.
- $S_i^{(a+)}$: the number of slots used at level i.
- P_a^+ : the level at which we poll, if the scheme is solved without polling, the variable obtains the value $n+1$.

We start with the delay analysis.

The Delay Analysis

To solve this problem we follow the same lines of reasoning as in Section 7.2.2. In this section we address the most significant differences with the evaluation in Section 7.2.2. Before going into the mathematical details, let us summarize the two major differences regarding the behaviour of the protocol. First, the scheme can no longer be solved before level S_l as these levels no longer exist. Secondly, polling at level S_l is no longer possible as level $S_l - 1$ is skipped.

(A+) Let us start with R_a^+. Notice that if the scheme was resolved at the first level S_l then it is also solved at or before level S_l with the ISA scheme, with a fixed starting level at 0, and vice versa. Secondly, the events $R_a^+ \leq x$ and $R_a \leq x$ coincide if $x > S_l$. Thus we have (due to (7.3))

$$P[R_a^+ = S_l \mid X_a^+ = k] = \frac{2^{(n-S_l)k} C_k^{2^{S_l}}}{C_k^{2^n}}, \tag{7.36}$$

$$P[R_a^+ \leq S_l + x \mid X_a^+ = k] = P[R_a \leq S_l + x \mid X_a = k], \tag{7.37}$$

for every value $x > 0$. This means that the probability of resolving the scheme before or at level S_l might decrease a bit, compared to the situation where the CC starts at level 0. If so, the probability that it is solved at level S_l+1 increases together with the probability of polling at this level. Remark that the probability of solving the scheme at level $S_l + 1$ without polling remains identical. There are no changes for the other levels. We define $p_a^+(k,i)$ as $P[R_a^+ = S_l + i - 1 \mid X_a^+ = k]$.

$\mathcal{F}_a^+(i,k)$ is defined as the probability that a tagged request is successful at or before level i. With similar arguments as used for R_a^+ we get

$$\begin{aligned}\mathcal{F}_a^+(S_l,k) &= \frac{C_{k-1}^{2^n-2^{n-S_l}}}{C_{k-1}^{2^n-1}}, &(7.38)\\ \mathcal{F}_a^+(S_l+x,k) &= \mathcal{F}_a(S_l+x,k), &(7.39)\end{aligned}$$

for every positive value x. The remainder of the analysis is analogue to the one with starting level zero.

The Throughput Analysis

The main objective of this section is to find the expected number of slots used at each level. Once we have these values, the distribution of X_a^+ allows us to calculate the throughput. This new scheme clearly never polls at level S_l (or before since these levels do not exist), therefore the probability of polling at level S_l+1 is increased. This causes the expected number of slots during level S_l and S_l+1 to be different from the ones we had before. All the other expected values remain the same. The expected number of slots at level S_l matches 2^{S_l} because we start at this level.

The situation for level S_l+1 is a bit more complicated. We start with equation (7.24) (where we add a '+' to all random variables and set i equal to S_l+1). In view of the discussion in **(A+)**, adding a '+' only changes the first two values (of the right-hand side) in this expression. Based on the fact that the events at level S_l are similar to those of the ISA scheme with the starting level at 0, the product of these two values is given by

$$\begin{aligned}&P[P_a^+ = S_l+1 \mid X_a^+ = k]E[S_{S_l+1}^{(a+)} \mid X_a^+ = k \cap P_a^+ = S_l+1] = \\ &\quad 2^{n-S_l}\sum_{i=1}^{u_{S_l}} iP[C_{S_l}^{(c)} = i \mid X_c = k]. \qquad (7.40)\end{aligned}$$

This concludes the throughput analysis.

7.2.4 Skipping the First Few Levels (DYNAMIC)

Having done the analysis for the static starting level it is easy to extend these results to the proposed dynamic model (see Section 6.4). We use the same random variables as above but substitute the '+' sign for a '$*$' to indicate the dynamic nature of the scheme. We also introduce a new random variable B^* as the starting level.

The Delay Analysis

We start with the search for R_a^* when conditioned on X_a^* and B^*. Assuming that the starting level equals S_l we have the following (due to the STATIC part):

$$\begin{aligned} P[R_a^* = S_l \mid X_a^* = k \cap B^* = S_l] &= \frac{2^{(n-S_l)k} C_k^{2^{S_l}}}{C_k^{2^n}}, && (7.41) \\ P[R_a^* \leq S_l + x \mid X_a^* = k \cap B^* = S_l] &= P[R_a \leq S_l + x \mid X_a = k], && (7.42) \end{aligned}$$

with x a positive number. Having found this we define $p_a^*(k, S_l, x+1)$ as $P[R_a^* = S_l + x \mid X_a^* = k \cap B^* = S_l]$. To find the joint distribution of (X_a^*, B^*) it is sufficient to construct the following transition matrix and to determine its steady state vector:

$$t_a^*(k; S_b, j; S_a) = P[X_{n+1}^{(a*)} = j \cap B_{n+1}^* = S_a \mid X_n^{(a*)} = k \cap B_n^* = S_b] =$$
$$\sum_{t=1}^{n+1-S_b} \frac{(\lambda t)^j e^{-\lambda t}}{j!} p_a^*(k, S_b, t) 1_{\{(t \leq B_l \wedge S_a = S_b - 1) \vee (B_l < t < B_m \wedge S_a = S_b) \vee (t \geq B_m \wedge S_a = S_b + 1)\}}.$$

Suppose that we observe the system at an arbitrary arrival instance, then the probability that this CC has a length of k frames and started at level S_l is needed. We also need the probability of having l contenders and a starting level S_l in the next CC (the CC that is preceded by the one containing the arbitrary arrival). These values are the natural extensions of (7.5) and (7.6)

$$P[X_{next}^{(a*)} = k \cap B_{next}^* = S_l] = \frac{P[X_a^* = k \cap B^* = S_l]k}{E[X_a^*]} \tag{7.43}$$

and

$$P[L_{cur}^{(a*)} = k \cap B_{cur}^* = S_l] = \frac{\lambda k}{E[X_a^*]} \sum_{j=1}^{2^n} P[X_a^* = j \ \cap \ B^* = S_l] p_a^*(j, S_l, k). \tag{7.44}$$

Finally, we need to find $\mathcal{F}_a^*(i, k, S_l)$, being the probability that a tagged arrival is successful at or before level i, knowing that there were $k-1$ other participants and the CC started at level S_l. Again, using the results of the previous section (see STATIC) we obtain

$$\begin{aligned} \mathcal{F}_a^*(S_l, k, S_l) &= \frac{C_{k-1}^{2^n - 2^{n-S_l}}}{C_{k-1}^{2^n - 1}}, \\ \mathcal{F}_a^*(S_l + x, k, S_l) &= \mathcal{F}_a(S_l + x, k). \end{aligned}$$

As before we can combine these results to obtain the average delay of the system. Let us now focus on the delay density function. As in (7.10), s is the number of frames that the tagged element competes, j is the length of the CC (in frames) in which the tagged request was generated and S_b the level at which this CC started. While $l-1$ is the number of other competitors next to the tagged request,

$$D_c(x) = \sum_{s=1}^{\lfloor x \rfloor} \sum_{S_b=S_{min}}^{S_{max}} \sum_{j=\lceil x \rceil - s}^{n+1} \sum_{l=1}^{2^n} \frac{\mathbf{\Delta}_1 \mathcal{F}_a^*(f(S_b, j) + s - 1, l, f(S_b, j))}{j} \cdot \\ \mathcal{G}_j(l)\ P[L_{cur}^{(a*)} = j\ \cap\ B_{cur}^* = S_b], \tag{7.45}$$

where $\mathcal{G}_j(l)$ was defined in Section 7.2.1, the function $f(S_b, j)$ is given by

$$f(S_b, j) = \begin{cases} \max(S_{min}, S_b - 1) & j \leq B_l \\ S_b & B_l < j < B_m \\ \min(S_{max}, S_b + 1) & j \geq B_m \end{cases}, \tag{7.46}$$

and $\mathbf{\Delta}_1 \mathcal{F}_a^*(x, y, z)$ equals $\mathcal{F}_a^*(x, y, z) - \mathcal{F}_a^*(x - 1, y, z)$.

The Throughput Analysis

In the section above we obtained the joint distribution of (X_a^*, B^*). This is used to derive the throughput T_a^* as follows:

$$T_a^* = \frac{E[X_a^*]}{\sum_{k=0}^{2^n} \sum_{S_l=S_{min}}^{S_{max}} P[X_a^* = k \cap B^* = S_l] E[\sum_i S_i^{(a*)} \mid X_a^* = k \cap B^* = S_l]}, \tag{7.47}$$

where the expected values were obtained in the evaluation of the static model.

7.2.5 Delay and Throughput for Multiple Instances

The analysis can also be used to evaluate the scenario with multiple instances. This is due to the fact that the delay experienced by a tagged request only depends upon the events happening in the instance it belongs to (because L is not taken into account).

7.3 Analysis of the Q-ary ISAP Protocol

In this section we generalize the analysis of the binary scheme to the Q-ary scheme. The work presented in this section was published in [65]. We demonstrate how the delay and throughput can be calculated for the Q-ary ISAP scheme if $S_l = 0$. The results for $S_l > 0$ can be obtained from those with $S_l = 0$. The procedure required to obtain the results for a higher starting level S_l is very similar for both the binary and the Q-ary case and therefore all details on this procedure are omitted. Also, results for ISA can be obtained by setting $N_p = 0$.

The following random variables will be used in the sequel of this section.

- X_c, resp. X_a, denotes the number of contenders or participants in a CC for Q-ary ISA, resp. ISAP.
- R_c, resp. R_a, denotes the level at which the CC is resolved (i.e., the number of frames needed minus one) for the Q-ary ISA scheme, resp. ISAP scheme.
- $C_i^{(c)}$ and $C_i^{(a)}$, denotes the number of collisions at level i for both protocols. These variables range from 0 to Q^i.
- P_a denotes the level at which we poll for the Q-ary ISAP scheme. If the scheme is solved without polling we let P_a be equal to $n+1$.

The symbol C_r^n is still used to denote the number of different possible combinations of r from n different items.

7.3.1 The Delay Analysis

Most of the steps presented below are straightforward generalizations of the binary equations, except for **(A2")**.

(A") We start by studying $P[R_a \leq i \mid X_a = k]$. Two cases can be considered: first, the CC might be solved before level i or at level i due to polling, secondly, the CC might be solved at level i without a switch to polling.

$$\begin{aligned} &P[R_a \leq i \mid X_a = k] = P[R_a \leq i-1 \cup P_a \leq i \mid X_a = k] + \\ &\quad P[R_a = i \cap P_a > i \mid X_a = k]. \end{aligned} \tag{7.48}$$

The first probability is discussed in **(A1")**, the second in **(A2")**.

(A1") We calculate the complementary probability mass. By definition of the polling mechanism (see Section 6.3) we have

$$P[R_a \geq i \cap P_a > i \mid X_a = k] = P[C_{i-1}^{(a)} > \left\lfloor \frac{N_p}{Q^{n-i+1}} \right\rfloor \mid X_a = k]. \tag{7.49}$$

The right-hand side is found using the following relationship:

$$P[C_{i-1}^{(a)} = \left\lfloor \frac{N_p}{Q^{n-i+1}} \right\rfloor + x \mid X_a = k] = P[C_{i-1}^{(c)} = \left\lfloor \frac{N_p}{Q^{n-i+1}} \right\rfloor + x \mid X_c = k], \tag{7.50}$$

for $x \geq 1$, but not necessarily for $x \leq 0$. The following variation on the Inclusion-Exclusion Principle [29, 73] is proposed (where the first equality is easily proven by induction on k):

$$s(i, Q^i, k) = \frac{Q^{(n-i)k} C_k^{Q^i}}{C_k^{Q^n}}, \tag{7.51}$$

$$s(i, l, k) = C_l^{Q^i} \sum_{l_1=0}^{l} Q^{(n-i)l_1} \frac{C_{l_1}^l C_{k-l_1}^{Q^n - lQ^{n-i}}}{C_k^{Q^n}} - \sum_{x=1}^{Q^i - l} C_l^{l+x} s(i, l+x, k), \tag{7.52}$$

where $s(i, l, k) = P[C_i^{(c)} = Q^i - l \mid X_c = k]$.

(A2") For the binary splitting algorithm this probability is found easily by observing that each collision at level $i-1$ involves only two MSs, otherwise it cannot be solved at level i. Clearly with Q-ary splitting this is no longer the case. Nevertheless, we still have the following equality:

$$\begin{aligned} P[C_{i-1}^{(a)} = \left\lfloor \frac{N_p}{Q^{n-i+1}} \right\rfloor + x \cap C_i^{(a)} = 0 \mid X_a = k] = \\ P[C_{i-1}^{(c)} = \left\lfloor \frac{N_p}{Q^{n-i+1}} \right\rfloor + x \cap C_i^{(c)} = 0 \mid X_a = k], \end{aligned} \tag{7.53}$$

for $x \geq 1$. At level i we can subdivide the address space in Q^i subsets of size Q^{n-i} based on the first i digits of the addresses. Each of these subsets is defined as a *virtual* slot at level i. We state that a *virtual* slot or subset at level i is collision free during a CC when there is at most one contender with an address that is part of that subset. Next, define $p(i, l_1, k)$ as the probability that at level i a specific set of l_1 *virtual* slots is collision free and that at level $i+1$ all *virtual* slots are collision free, given that we had k contenders in the CC. Notice that the number of collisions at level i might be smaller than $Q^i - l_1$, so other *virtual* slots that do not belong to the specific set of size l_1 might also be collision free. Hence,

$$p(i, l_1, k) = \frac{1}{C_k^{Q^n}} \sum_{j=0}^{l_1} Q^{(n-i)j} C_j^{l_1} C_{k-j}^{Q^{i+1} - l_1 Q} Q^{(n-i-1)(k-j)}. \tag{7.54}$$

Next, define $q(i, l_1, k)$ as the probability that level i contains $Q^i - l_1$ collisions and level $i+1$ is collision free. Then, we have the following relationship between $p(i, l_1, k)$ and $q(i, l_1, k)$:

$$q(i, Q^i, k) = p(i, Q^i, k), \tag{7.55}$$

$$q(i, l_1, k) = C_{l_1}^{Q^i} p(i, l_1, k) - \sum_{x=1}^{Q^i - l_1} C_{l_1}^{l_1+x} q(i, l_1 + x, k). \tag{7.56}$$

This completes **(A2")**.

(B") X_a is the steady-state vector of the Markovian process $(X_n^{(a)})_n$, where $X_n^{(a)}$ denotes the number of contenders during the n-th CC. Due to **(A")**,

$$t_a(k, j) \overset{\text{def}}{=} P[X_{n+1}^{(c)} = j \mid X_n^{(c)} = k] = \sum_{t=1}^{n+1} \frac{(\lambda t)^j e^{-\lambda t}}{j!} P[R_a = t - 1 \mid X_a = k], \tag{7.57}$$

for $0 \leq j \leq Q^n - 1$. For $j = Q^n$ we assign the remaining probability mass. X_a is then found by solving the related eigenvector problem.

(C") Observing the system at an arbitrary arrival instance O_n, we require the probability that the length of the CC, that contains O_n, is k frames and that there are l contenders in the next CC. We denote $L_{cur}^{(a)}$ and $X_{next}^{(a)}$ as the length of the CC containing O_n and the number of participants in the next CC. Some straightforward reasoning shows the following relationship between $X_{next}^{(a)}$, $L_{cur}^{(a)}$ and X_a:

$$P[X_{next}^{(a)} = l] = \frac{P[X_a = l]l}{E[X_a]}, \tag{7.58}$$

and

$$P[L_{cur}^{(a)} = l] = \lambda l \sum_{k=0}^{Q^n} P[X_a = k] P[R_a = l - 1 \mid X_a = k]/E[X_a]. \tag{7.59}$$

(D") Define $\mathcal{F}_a(i,k)$ as the probability that a tagged request is successful at or before level i given that we had k contenders in the CC (for ISAP). Next, define $v_i = 1 + \lfloor \frac{N_p}{Q^{n-i+1}} \rfloor$. We have

$$\mathcal{F}_a(i,k) = P[R_a \leq i - 1 \cup P_a \leq i \mid X_a = k] + \sum_{s \geq v_i} P[R_t \leq i \cap C_{i-1}^{(c)} = s \mid X_c = k], \tag{7.60}$$

where R_t denotes the level at which our tagged request is successful. The first probability was found in **(A1")**. The second one is calculated using a similar method as in (7.51) and (7.52). We define $t(i,s,k)$ as $P[R_t \leq i \cap C_{i-1}^{(c)} = Q^{i-1} - s \mid X_c = k]$. Then we get (where the first equation is a consequence of (7.51))

$$t(i, Q^{i-1}, k) = \frac{Q^{(n-i+1)k} C_k^{Q^{i-1}}}{C_k^{Q^n}} \tag{7.61}$$

$$\begin{aligned} t(i,s,k) = & -\sum_{x=1}^{Q^{i-1}-s} C_s^{s+x} t(i, s+x, k) + C_s^{Q^{i-1}} \sum_{l_1=0}^{s} Q^{(n-i+1)l_1} \cdot \\ & \frac{C_{l_1}^s C_{k-l_1}^{Q^n - sQ^{n-i+1}}}{C_k^{Q^n}} \left(\frac{l_1}{k} + \left(1 - \frac{l_1}{k}\right) \frac{C_{k-l_1-1}^{Q^n - sQ^{n-i+1} - Q^{n-i}}}{C_{k-l_1-1}^{Q^n - sQ^{n-i+1} - 1}} \right). \end{aligned} \tag{7.62}$$

With these values it is straightforward to find the second term of expression (7.60).

(A",B",C",D") Having done this we can calculate the mean delay. The delay can be split into two parts. The first D_1 is the time until the start of the next CC and the second D_2 is the number of frames needed until our tagged request is successful. Using expression (7.59) and knowing that the arrivals are distributed uniformly within a CC (see Section 7.1), the expected value for the first part equals

$$E[D_1] = \sum_{i=1}^{n+1} P[L_{cur}^{(a)} = i] i/2. \tag{7.63}$$

By definition of the expected value the second part equals

$$E[D_2] = \sum_{i=0}^{n} \sum_{k \geq 1} P[X_{next}^{(a)} = k](i+1)(\mathcal{F}_a(i,k) - \mathcal{F}_a(i-1,k)), \tag{7.64}$$

where $\mathcal{F}_a(i,k)$ was defined in **(D")**. The delay density function $D_a(x)$ (with x between 1 and $2(n+1)$) is the following:

$$D_a(x) = \sum_{s=1}^{\lfloor x \rfloor} \sum_{j=\lceil x \rceil - s}^{n+1} \sum_{l=1}^{Q^n} \frac{\mathcal{F}_a(s-1,l) - \mathcal{F}_a(s-2,l)}{j} \mathcal{G}_j(l) P[L_{cur}^{(a)} = j]. \tag{7.65}$$

where $\mathcal{G}_j(l), 1 \leq l \leq Q^n$ is a probability distribution that is equal to $\frac{(\lambda j)^{l-1}}{(l-1)!} e^{-\lambda j}$ for $l < Q^n$ (the remaining probability mass is assigned to $\mathcal{G}_j(Q^n)$). In equation (7.65), s denotes the number of transmissions (including the successful transmission) a tagged request needs and j refers to the length (in frames) of the CC in which our tagged request is generated. Finally, $l-1$ equals the number of other competitors in the CC apart from our tagged one.

7.3.2 The Throughput Analysis

The throughput analysis for the Q-ary ISAP scheme is very different from the one used to obtain the throughput of the binary scheme. Although it is possible to use the same method as in the binary case, we opt for a shorter but numerically more sensitive method. The vital part of this method is to calculate the joint probability distribution of the number of collision at level i and level $i+1$ for the ISA scheme. These probabilities are calculated in a numerically exact environment (Mathematica).

Define a new set of random variables $S_i^{(a)}$, where $S_i^{(a)}$ is the number of slots required at level i when using the ISAP scheme. By definition of the throughput T_a we have that

$$T_a = \frac{E[X_a]}{\sum_{k=0}^{Q^n} P[X_a = k] E[\sum_i S_i^{(a)} \mid X_a = k]}. \tag{7.66}$$

As the probabilities $P[X_a = k]$ were obtained during the delay analysis, it is sufficient to find $E[\sum_i S_i^{(a)} \mid X_a = k]$. The expected number of slots used during a CC equals the sum of the expected number of slots used at each level, we can focus on $E[S_i^{(a)} \mid X_a = k]$. Some preliminary calculations are presented in **(E")** (in **(E")**, N_p is equal to zero) and in **(F")** we calculate $E[S_i^{(a)} \mid X_a = k]$ using the results of **(E")**.

(E") Define $p(i, l_1, l_2, k)$ to be the probability that, at level i, a specific collection of l_1 *virtual* slots is collision free and, at level $i+1$, there are exactly l_2 collision free *virtual* slots given that we had k contenders in the CC. The definition of a *virtual* slot was presented in **(A2")**. Notice that the number of collisions at level i might be smaller than $Q^i - l_1$; thus, other *virtual* slots that do not belong to the specific collection of size l_1 might also be collision free. A reasoning based on the Inclusion-Exclusion Principle [29, 73] allows us to state the following:

$$\begin{aligned} p(i,l_1,l_2,k) &= \frac{1}{C_k^{Q^n}} \sum_{j=0}^{l_1} Q^{(n-i)j} C_j^{l_1} C_s^{Q^{i+1}-Ql_1} \sum_{j'=0}^{s} Q^{(n-i-1)j'} C_{j'}^{s} C_{k-j-j'}^{Q^n-l_1Q^{n-i}-sQ^{n-i-1}} \\ &\quad - \sum_{x=1}^{Q^{i+1}-Ql_1-s} C_s^{s+x} p(i,l_1,l_2+x,k), \end{aligned} \tag{7.67}$$

with $s = l_2 - Ql_1$ and with $p(i,l_1,l_2,k) = 0$ for $l_2 < Ql_1$. Next, we define $s(i,l_1,l_2,k)$ as the probability of having exactly l_1 collision free *virtual* slots, at level i, and exactly l_2 collisions free *virtual* slots, at level $i+1$, given that we had k contenders in the CC. We have the following relationship between $p(i,l_1,l_2,k)$ and $s(i,l_1,l_2,k)$:

$$s(i,Q^i,l_2,k) = p(i,Q^i,l_2,k), \tag{7.68}$$

$$s(i,l_1,l_2,k) = C_{l_1}^{Q^i} p(i,l_1,l_2,k) - \sum_{x=1}^{Q^i-l_1} C_{l_1}^{l_1+x} s(i,l_1+x,l_2,k). \tag{7.69}$$

This concludes part **(E”)**.

(F”) Since the expected number of slots at level 0 and 1 are straightforward to obtain, we can focus on $E[S_i^{(a)} \mid X_a = k]$ for $i \geq 2$. We distinguish between the following three events $E_1^{(i)}, E_2^{(i)}$ and $E_3^{(i)}$:

- $E_1^{(i)}$: the CC is resolved within the first $i-2$ levels (with or without polling) or polling takes place at level $i-1$.
- $E_2^{(i)}$: the CC is resolved (without polling) at level $i-1$ or polling takes place at level i.
- $E_3^{(i)}$: the CC is not resolved within the first $i-1$ levels and polling does not occur at level i.

Thus, $E[S_i^{(a)}] = P(E_1^{(i)})E[S_i^{(a)} \mid E_1^{(i)}] + P(E_2^{(i)})E[S_i^{(a)} \mid E_2^{(i)}] + P(E_3^{(i)})E[S_i^{(a)} \mid E_3^{(i)}]$. Provided that the first event $E_1^{(i)}$ occurs, the expected number of slots $S_i^{(a)}$ at level i equals zero. As for the other two, we can rewrite the previously mentioned events as: $E_1^{(i)} = C_{i-2}^{(c)} < v_{i-1}$, $E_2^{(i)} = C_{i-2} \geq v_{i-1} \cap C_{i-1} < v_i$ and $E_3^{(i)} = C_{i-2} \geq v_{i-1} \cap C_{i-1} \geq v_i$ (v_i was defined in **(D”)**). Moreover,

$$\begin{aligned} &P(E_2^{(i)} \mid X = k)E[S_i^{(a)} \mid X = k \cap E_2^{(i)}] = \\ &\qquad \sum_{l_1 \geq v_{i-1}} \sum_{l_2 < v_i} s(i-2, Q^{i-2}-l_1, Q^{i-1}-l_2, k)\ Q^{n-i+1}\ l_2, \end{aligned} \tag{7.70}$$

and finally

$$P(E_3^{(i)} \mid X = k)E[S_i^{(a)} \mid X = k \cap E_3^{(i)}] = \\ \sum_{l_1 \geq v_{i-1}} \sum_{l_2 \geq v_i} s(i-2, Q^{i-2} - l_1, Q^{i-1} - l_2, k)\ Q\ l_2, \tag{7.71}$$

where $s(i, l_1, l_2, k)$ was found in **(E")**.

7.4 Analysis of the Optional Parameter M_p

For the definition and use of the parameter M_p we refer to Section 6.7. As a reminder, the ISAP scheme that uses the M_p parameter is referred to as the M-ISAP scheme. The following random variables will be used in the sequel of this section.

- X_c, X_a and $\tilde{X}_a$ denote the number of contenders or participants in a CC for the ISA, ISAP and M-ISAP scheme.
- R_c, R_a and $\tilde{R}_a$ denote the level at which the CC is resolved (i.e., the number of frames needed minus 1) for ISA, ISAP and M-ISAP.
- $C_i^{(c)}$, $C_i^{(a)}$ and $\tilde{C}_i^{(a)}$, denote the number of collisions at level i for each scheme. These variables range from 0 to Q^i.
- $S_i^{(c)}$, $S_i^{(a)}$ and $\tilde{S}_i^{(a)}$ denote the number of contention slots at level i for each scheme.
- P_a and $\tilde{P}_a$ denote the level at which we poll for the ISAP and M-ISAP scheme. If a CC is solved without polling we let P_a and $\tilde{P}_a$ be equal to $n+1$.

Furthermore, we use the symbol C_r^n to denote the number of different possible combinations of r from n different items.

7.4.1 Delay and Throughput Analysis

The influence of the parameter M_p on the performance can be studied by introducing some modifications to the original analysis of the ISAP as performed in Sections 7.2 and 7.3. In this section we summarize the main modifications required. Most of the modifications required make use of the following property. Consider the i-th level of a CC with k contenders. If $i < M_p$ then both ISA and M-ISAP behave identical (polling is not allowed at these level by definition of M_p). For $i > M_p$ we get an identical behavior for ISAP and M-ISAP. For $i = M_p$ the M-ISAP scheme behaves differently from both the ISA and ISAP scheme.

The following three sets of equations are all due to this property. First, $\tilde{R}_a$ when conditioned on $\tilde{X}_a$ can be calculated as follows:

$$P[\tilde{R}_a \leq i \mid \tilde{X}_a = k] = P[R_c \leq i \mid X_c = k] \qquad i < M_p, \tag{7.72}$$
$$P[\tilde{R}_a \leq i \mid \tilde{X}_a = k] = P[R_a \leq i \mid X_a = k] \qquad i \geq M_p. \tag{7.73}$$

Second, define $\mathcal{F}_c(i,k), \mathcal{F}_a(i,k)$ and $\tilde{\mathcal{F}}_a(i,k)$ as the probability that a tagged station is successful after at most $i+1$ transmissions provided that the CC had k contenders for each of the schemes. Then,

$$\tilde{\mathcal{F}}_a(i,k) = \mathcal{F}_c(i,k) \qquad i < M_p, \tag{7.74}$$
$$\tilde{\mathcal{F}}_a(i,k) = \mathcal{F}_a(i,k) \qquad i \geq M_p. \tag{7.75}$$

Finally, we also have

$$E[\tilde{S}_i^{(a)} \mid \tilde{X}_a = k] = E[S_i^{(c)} \mid X_c = k] \qquad i < M_p, \tag{7.76}$$
$$E[\tilde{S}_i^{(a)} \mid \tilde{X}_a = k] = E[S_i^{(a)} \mid X_a = k] \qquad i > M_p. \tag{7.77}$$

The first two modifications are sufficient to calculate the average delay and the delay density function for the M-ISAP scheme. As for the throughput, we still need to obtain $E[\tilde{S}_{M_p}^{(a)} \mid \tilde{X}_a = k]$. Consider a CC with k contenders, we mentioned that the M-ISAP scheme behaves identical to ISA until level $M_p - 1$. Therefore,

$$E[\tilde{S}_{M_p}^{(a)} \mid \tilde{X}_a = k] = \sum_{j \leq \lfloor N_p / Q^{n-M_p+1} \rfloor} P[C_{M_p-1}^{(c)} = j \mid X_c = k]\; Q^{n-M_p+1}\; j \; + \sum_{j > \lfloor N_p / Q^{n-M_p+1} \rfloor} P[C_{M_p-1}^{(c)} = j \mid X_c = k]\; Q\; j. \tag{7.78}$$

An algorithm based on the Inclusion-Exclusion Principle [29, 73] to calculate $P[C_i^{(c)} = j \mid X_c = k]$ was provided in the previous sections.

7.5 Analysis of the Impact of L

In the previous two sections we calculated the delay distribution and the throughput of the ISAP scheme where L, the maximum number of contention slots allowed in a single frame, was not taken into account. In this section we focus on the parameter L. Unfortunately, it seems like there is no (apparent) practical way to calculate the throughput and delay distribution of ISAP when L is taken into account. Theoretically it is not too difficult to design an algorithm that calculates the throughput and delay of ISAP. However, the time and space complexity of the algorithm is too large. Also, the calculations are numerically sensitive and have to be conducted in a numerically exact environment (using rational calculations).

The main bottleneck is to obtain the probability that a CC requires $i+1$ frames (provided we have k participants). In the previous two sections these probabilities were closely related with $P[R_a = i \mid X_a = k]$, but this is no longer the case if a level requires multiple frames. It is however possible, by extending the method we used to obtain the joint distribution of the number of collisions at level i and $i+1$ (see Section 7.3.2), to set up an algorithm that calculates the joint distribution of the number of collisions (of ISA) at level $0, 1, \ldots, n$, from which it is easy to obtain the above-mentioned probabilities. However, the amount of memory required to store this joint distribution is huge: $Q^{n(n+1)/2}$ probabilities have to be stored. Therefore, for realistic values of n, this algorithm is out of reach of the current computer generations, but might become realistic in 10 years time ☺.

Instead of waiting for another 10 years, we can however calculate some measures, i.e., expected values, that are closely related to the delay and the throughput of ISAP. These measures provide insight on the interaction between the parameter L and the other protocol parameters: N_p and S_l. We restrict ourselves to the binary case $Q = 2$. The same technique can also be used for $Q > 2$. This work was published in [67]

7.5.1 Delay and Throughput Measures

As a reminder, let us summarize the following important protocol parameters:

- n : the length of the MAC addresses (in bits).
- L : the maximum number of contention slots allowed in one frame.
- N_p : the value that triggers the polling mechanism.
- S_l : the starting level.

In this section we calculate the following expected values:

- $E[F \mid X = k]$: the expected length of a CC (expressed in frames) with $k \geq 2$ participants.
- $E[S \mid X = k]$: the expected number of contention slots that a CC with $k \geq 2$ participants requires.

The value $E[F]$ is strongly related with the delay experienced by the protocol, whereas $E[S]$ is related with the throughput of the protocol. Also, notice that $E[S \mid X = k]$ does not depend upon the value of L. Moreover, in the special case of $L = 1$ both expected values ($E[F]$ and $E[S]$) are identical. Therefore, it is sufficient to set up a scheme to calculate $E[F \mid X = k]$ for any value of L.

The first step of the calculation is identical to **(E")** (see Section 7.3.2), where we calculate the probabilities $s(i, l_1, l_2, k)$ of having l_1 collision free *virtual* slots at level i and l_2 collision free *virtual* slots at level $i+1$ provided that we had k contenders in the CC. See **(A2")** in

Section 7.3.1 for the definition of a *virtual* slot. The remainder of the analysis is divided into two parts: in the first part $N_p \geq 0$ and $S_l = 0$, while in the second case $N_p \geq 0$ and $S_l \geq 0$.

Part 1: $0 \leq N_p < 2^n$ **and** $S_l = 0$

Define the random variable F as the number of frames required to support a CC and the random variable F_i as the number of frames required to support level i of the tree, then

$$E[F \mid X = k] = \sum_{i=0}^{n} E[F_i \mid X = k]. \tag{7.79}$$

For $k \geq 2$ and $N_p < 2^n$, we have $F_0 = 1$ and $F_1 = 1$, resp. 2, if $L \geq 2$, resp. $L = 1$. Therefore, we can focus on $E[F_i \mid X = k]$ with $i \geq 2$. We separate the following three events $E_1^{(i)}, E_2^{(i)}$ and $E_3^{(i)}$:

- $E_1^{(i)}$: the CC is resolved within the first $i-2$ levels (with or without polling) or polling takes place at level $i-1$.
- $E_2^{(i)}$: the CC is resolved (without polling) at level $i-1$ or polling takes place at level i.
- $E_3^{(i)}$: the CC is not resolved within the first $i-1$ levels and polling does not occur at level i.

Thus, $E[F_i] = P(E_1^{(i)})E[F_i \mid E_1^{(i)}] + P(E_2^{(i)})E[F_i \mid E_2^{(i)}] + P(E_3^{(i)})E[F_i \mid E_3^{(i)}]$. Given that the first event $E_1^{(i)}$ occurs, the expected number of frames F_i at level i equals zero. The two other expressions are found as follows.

Define C_i as the number of collisions at level i. Suppose that $C_i = N_c$, then the size of the remaining address space is $N_c 2^{n-i}$. Thus, at level $i+1$ we have no polling when $N_c > N_p/2^{n-i}$. Also, having $N_c > N_p/2^{n-i}$ is equivalent to having $N_c > \lfloor N_p/2^{n-i} \rfloor$ for N_c an integer value. Hence, polling does not occur at level $i+1$ if $C_i \geq 1 + \lfloor N_p/2^{n-i} \rfloor$. We denote $1 + \lfloor N_p/2^{n-i} \rfloor$ as c_i. Hence, we can rewrite the previously mentioned events as: $E_1^{(i)} = C_{i-2} < c_{i-2}$, $E_2^{(i)} = C_{i-2} \geq c_{i-2} \cap C_{i-1} < c_{i-1}$ and $E_3^{(i)} = C_{i-2} \geq c_{i-1} \cap C_{i-1} \geq c_{i-1}$. We already mentioned that $E[F_i \mid X = k \cap E_1^{(i)}]$ is zero. Also,

$$P(E_2^{(i)} \mid X = k)E[F_i \mid X = k \cap E_2^{(i)}] = \sum_{l_1 \geq c_{i-2}} \sum_{l_2 < c_{i-1}} s(i-2, 2^{i-2} - l_1, 2^{i-1} - l_2, k) \left\lceil \frac{2^{n-i+1} l_2}{L} \right\rceil, \tag{7.80}$$

and finally

$$P(E_3^{(i)} \mid X = k)E[F_i \mid X = k \cap E_3^{(i)}] = \sum_{l_1 \geq c_{i-2}} \sum_{l_2 \geq c_{i-1}} s(i-2, 2^{i-2} - l_1, 2^{i-1} - l_2, k) \left\lceil \frac{2 l_2}{L} \right\rceil, \tag{7.81}$$

where $s(i, l_1, l_2, k)$ was found in **(E")** (see Section 7.3.2).

Part 2: $0 \leq N_p < 2^n$ **and** $S_l \geq 0$

To avoid any confusion with the previous we define $F_i(S_l)$ as the number of frames required to support level i knowing that the starting level is S_l. Clearly, for $x < S_l$ and $y > S_l + 1$

$$\begin{aligned} E[F_x(S_l) \mid X = k] &= 0, && (7.82) \\ E[F_{S_l}(S_l) \mid X = k] &= \left\lceil \frac{2^{S_l}}{L} \right\rceil, && (7.83) \\ E[F_y(S_l) \mid X = k] &= E[F_y(0) \mid X = k], && (7.84) \end{aligned}$$

where $E[F_i(0) \mid X = k]$ was found in part 1. Thus, only the expected number of frames to support level $S_l + 1$ remains to be determined. We separate three events:

- $E_1(S_l)$: the CC is solved at level S_l.
- $E_2(S_l)$: polling occurs at level $S_l + 1$.
- $E_3(S_l)$: the CC is not solved at level S_l nor does polling occur at level $S_l + 1$.

Making use of the values c_i defined in Part 1, we can rewrite these events as $C_{S_l} = 0$, $C_{S_l} > 0 \cap C_{S_l} < c_{S_l}$ and $C_{S_l} \geq c_{S_l}$. Hence,

$$E[F_{S_l+1}(S_l) \mid X = k] = \sum_{l_1} \left(\sum_{l_2 < c_{S_l}} s(S_l - 1, l_1, 2^{S_l} - l_2, k) \left\lceil \frac{2^{n-S_l} l_2}{L} \right\rceil + \sum_{l_2 \geq c_{S_l}} s(S_l - 1, l_1, 2^{S_l} - l_2, k) \left\lceil \frac{2 l_2}{L} \right\rceil \right),$$

for $S_l > 0$. The results for $S_l = 0$ were obtained in Part 1 of the analysis.

Chapter 8

Results for the Identifier Splitting Algorithm combined with Polling

This chapter investigates the influence of the different ISAP protocol parameters. Our main objective is to obtain a well-founded understanding of the impact of the different protocol parameters on the delay and throughput characteristics and to reveal possible delay vs. throughput tradeoffs. Petras, *et al* [50–52] have calculated the first two moments of the length of an ISA CC, with k contenders. From these values they estimated the mean delay and throughput of ISA by assuming that a station generates a new arrival during a CC with probability p. Thus, the number of arrivals occurring during a CC obeys a binomial distribution and is independent of the length of a CC. This assumed independence results in mean delay and throughput results that are (far) too optimistic.

This chapter is subdivided into five sections. Section 8.1 presents some numerical examples for the binary ISAP scheme. In Section 8.2 we investigate the impact of the splitting factor Q. Section 8.3 demonstrates the influence of the optional parameter M_p, while Section 8.4 focusses on the parameter L. Finally, in Section 8.5, we summarize the main conclusions.

8.1 Results for the Binary ISAP Scheme

In this section we use the analytical model, presented in Section 7.2, to investigate the impact of the arrival rate λ, the trigger value N_p and the starting level S_l on the mean delay, the delay density function and the throughput for the binary ISAP scheme. The system parameters are set as follows. The number of mobiles is 128; that is, $n = 7$. The arrival rate λ (requests per frame) varies between 0.05 and 3.5. The values studied for the polling threshold N_p are 0, 20 and 40, where the first case corresponds with the ISA scheme. The starting level S_l will vary from level 0 to 2. When studying a system with a dynamic starting level, B_l and B_m are set to 1 and 4 respectively. Therefore, the starting level is decreased by one if the CC is solved in 1 frame and is increased by one if the CC consist of 4 or more frames. The boundary values are set as follows: $S_{min} = 0$ and $S_{max} = 2$. The number of instances varies between 1 and 4.

We study four different scenarios. First, we investigate the impact of the polling threshold N_p, when the starting level S_l is fixed at 0. Next, the influence of the starting level S_l is discussed. Then, the impact of using a variable starting level is considered. Finally, we look at the effect of using multiple instances of ISA. Additional numerical results can be found in [72].

8.1.1 The Influence of the Polling Threshold on the System Performance

Figures 8.1 and 8.2 show the influence switching to polling has on the mean delay and the throughput. As expected we get a tradeoff between the delay and throughput characteristics: the sooner the ISAP protocol switches to polling, the shorter the mean delay, but the lower the throughput.

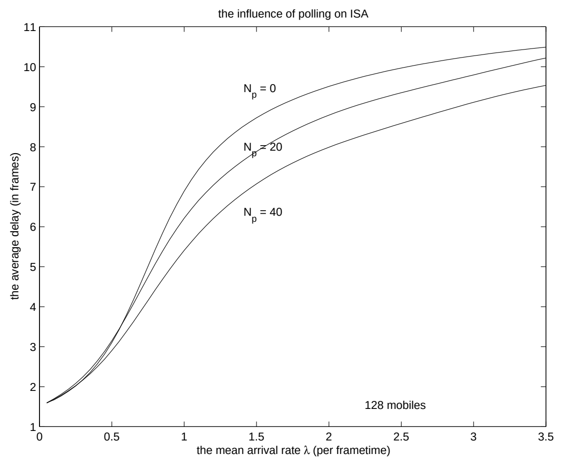


Figure 8.1: *The impact of polling on the mean delay*

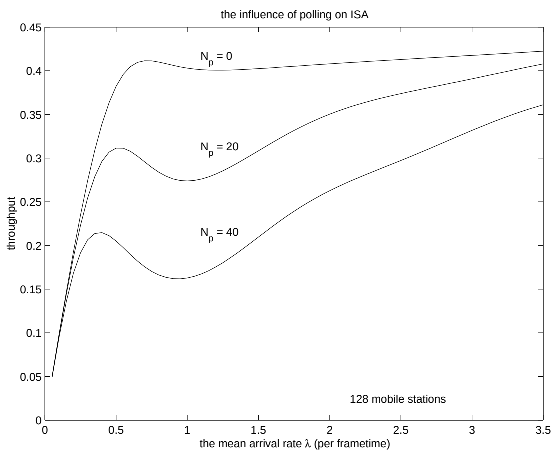


Figure 8.2: *The impact of polling on the throughput*

From Figures 8.1 and 8.2 we observe that the protocol behaves very similar for different N_p values when the arrival rate λ is small (below 0.25). A similar result is obtained for large values of λ (beyond 5). Both these results are intuitively clear. Polling is not an issue in these cases: for λ very small, collisions rarely occur and are solved before polling can be considered; if λ is very large, the remaining size of the address space is too large to switch to polling.

Let us now consider moderate values for λ. Recall that for a polling threshold $N_p = 40$, resp. $N_p = 20$, the protocol will never start polling until level 3, resp. level 4 (a single collision at level i corresponds to a remaining address space of 2^{7-i}). Thus, the impact of N_p on the performance measures is low for small values of λ. If the arrival rate increases (look at the range 0.5 till 1), the probability that collisions at level 2, resp. 3 are introduced increases. In most cases these collisions contain very few participants; that is, occasionally 32, resp. 16, polling slots are provided at level 3, resp. 4, to poll very few competitors. Therefore, the throughput decreases with increasing values of N_p. If λ is increased even more, beyond one, polling is postponed in most cases to a later level (as the expected number of collisions at level 2, resp. 3, becomes larger than 1) and will contain more participants. This results in higher throughput values for a fixed value of N_p.

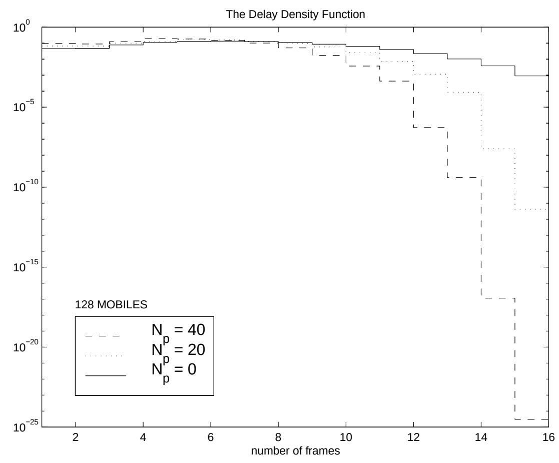


Figure 8.3: *The impact of polling on the delay density function with $\lambda = 1$*

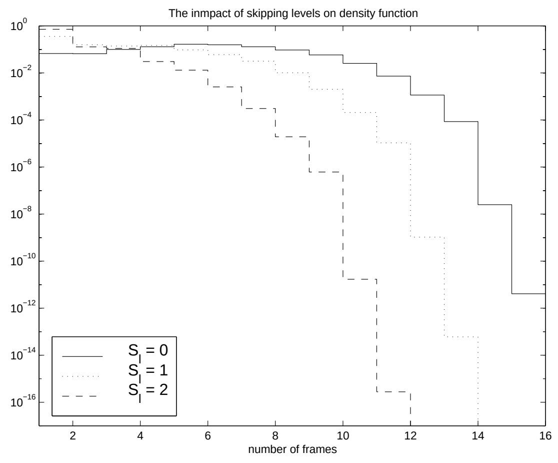


Figure 8.4: *The impact of Skipping with $\lambda = 1$ and $N_p = 20$*

Figure 8.3 shows the impact of polling on the delay density function (for $\lambda = 1$). It illustrates that the main improvement of the delay is located within the tail of the distribution and is not merely an improvement of the mean delay.

8.1.2 The Influence of Skipping Levels (STATIC) on the System Performance

Figures 8.5 and 8.6 illustrate the impact of S_l on the average delay and the throughput. In these figures we have three different types of curves: full, dotted and dashed, corresponding to $S_l = 0, 1$ and 2 respectively. Moreover, for each value of S_l the results for $N_p = 0, 20$ and 40 are depicted. For a fixed value of S_l, the upper curve, in both the delay and throughput results, corresponds to $N_p = 0$, the middle curve to $N_p = 20$ and the lowest to $N_p = 40$.

Skipping the first levels leads to a decrease of the mean waiting time. Let us focus on the impact of polling for variable values of S_l. First, Figure 8.5 shows a larger decrease of the delay due to polling, when the starting level is larger. This can be seen by observing the area between the curves for $N_p = 0, 20$ and 40. Secondly, notice that the curves converge slower for increasing values of S_l (observe the differences for $\lambda = 3.5$ in Figure 8.5). Figure 8.6 represents the throughput results for $N_p = 0, 20$ and 40. We see that, for low values of λ, skipping levels results in a lower throughput (as most of the 2^{S_l} slots are wasted). If λ becomes larger, this loss is converted in a small gain due to the fact that the majority of the slots before level S_l contains collisions. The influence of skipping levels on the delay density function is shown in Figure 8.4.

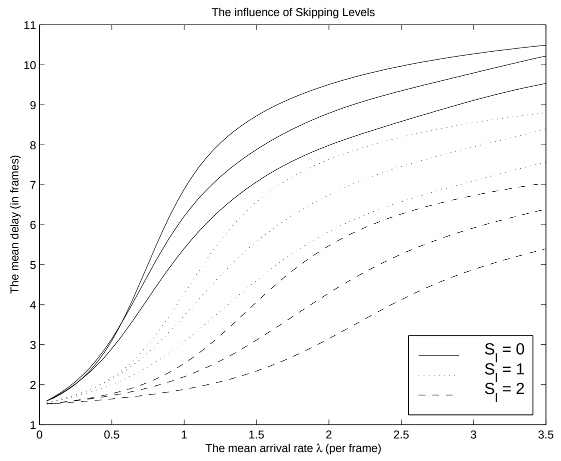


Figure 8.5: *The influence of skipping on the mean delay.*

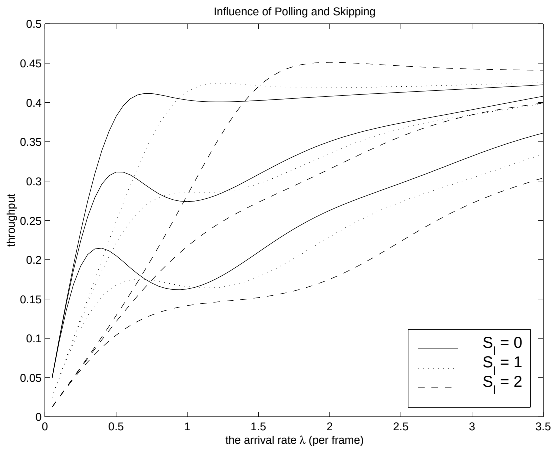


Figure 8.6: *The influence on the throughput for N_p = 0, 20 and 40.*

8.1.3 The Influence of Skipping Levels (DYNAMIC) on the System Performance

From the previous sections we may conclude that a higher starting level has a positive impact on the delay and even on the throughput, especially for larger values of λ. Unfortunately a high price is paid for this in terms of throughput if λ is small. The aim of this section is to show that the dynamic scheme as proposed in Section 6.4 solves this problem. That is, if λ is small the results should tend to the results for $S_l = S_{min}$, while for λ large the behavior should be similar to the one corresponding to $S_l = S_{max}$. Figures 8.7 and 8.8 show that this is the case (for $N_p = 20$), meaning that a system where the levels are skipped dynamically, is able to limit the maximum delay while keeping the throughput high.

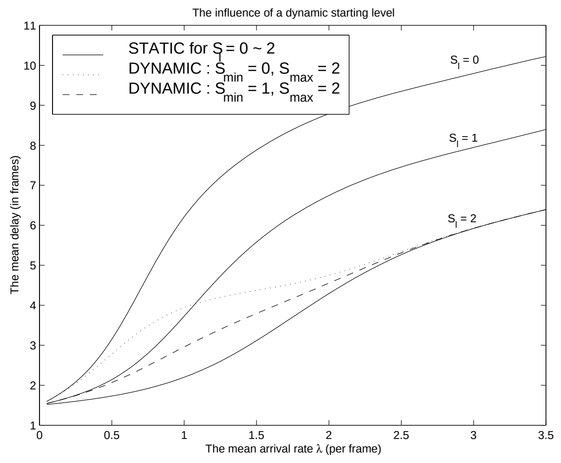


Figure 8.7: *The influence of dynamic skipping on the delay ($N_p = 20$).*

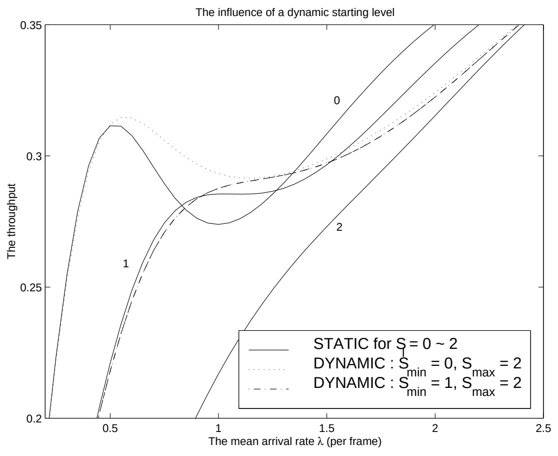


Figure 8.8: *The influence of dynamic skipping on the throughput for $N_p = 20$.*

8.1.4 The influence of Multiple Instances of ISA on the System Performance

The analysis presented in Section 7.2 can be applied in order to evaluate the influence of multiple instances. In this final scenario λ varies between 0 and 6. Figures 8.9 and 8.10 show the delay and throughput results for three configurations. In the first, we have one instance and the starting level S_l is fixed at 2. In the second, we have two instances, with $S_l = 1$. Finally, we have four instances, with $S_l = 0$.

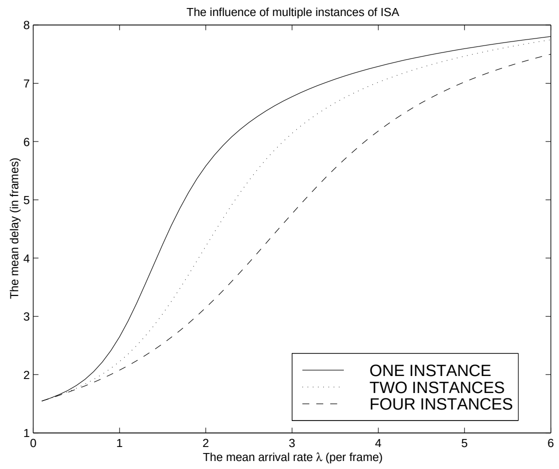


Figure 8.9: *The influence of multiple instances on the delay.*

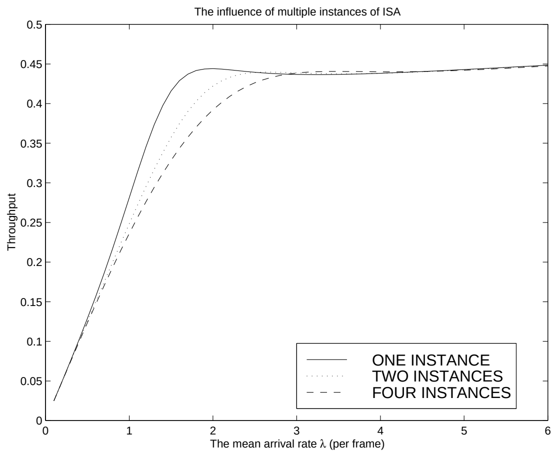


Figure 8.10: *The influence of multiple instances on the throughput.*

Clearly, the more instances we use, the better the average delay. Except for very small and very large values of λ, were all scenarios perform alike. For more moderate values of λ, there exists a tradeoff between the delay and throughput; thus, the more instances we use, the smaller the delay and the lower the throughput. Still, the decrease in throughput is considerably smaller, compared to the throughput losses caused by the introduction of N_p, thereby making the use of multiple instances attractive.

8.2 Results for the Q-ary ISAP Scheme

In this section, we investigate the influence of the splitting factor Q and its interaction with the arrival rate λ, the trigger value N_p and, to some extent, the starting level S_l. Moreover, we check whether the main conclusions for the binary ISAP scheme are still valid for the Q-ary scheme. The system parameters are set as follows. The splitting factor Q equals $2, 3$ or 4, we refer to these three cases as the binary, ternary and quaternary scheme. The number of digits n depends upon the value of Q. In the binary case n equals 8, in the ternary case n equals 5 and finally in the quaternary case n equals 4. Thus, for the binary and quaternary scheme we are able to support 256 MSs, in the ternary case we can have at most 243 MSs. This small difference in the size of the address space should hardly have any effect on the results because on average the number of participating MSs in a CC is always much smaller than Q^n.

8.2.1 The Influence of the Splitting Factor and the Polling Threshold on the System Performance

In Figures 8.11 and 8.12, the influence of Q on the mean delay and the delay density function is shown for $N_p = 0$ and $N_p = 20$. First, a larger splitting factor Q results in a smaller delay (mean and quantiles). Also, the delay differs much more when we compare the binary and ternary scheme as opposed to the ternary and quaternary scheme. In general, a larger value for Q results in a smaller delay. Also, the delay improvement we get from increasing Q by one decreases as Q grows. Indeed, increasing Q by one results in $1/Q$ times as many slots to resolve a collision.

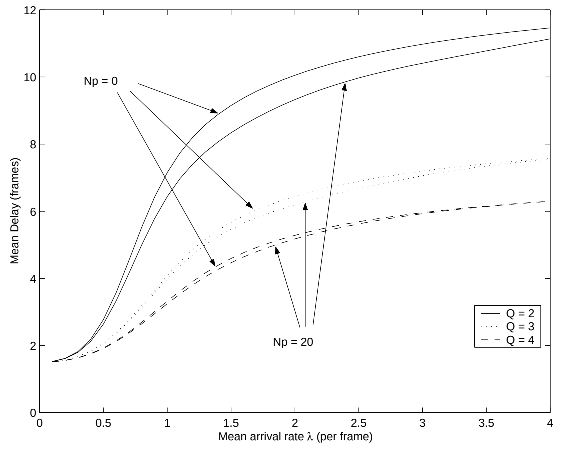


Figure 8.11: *The impact of Q and N_p on the mean delay*

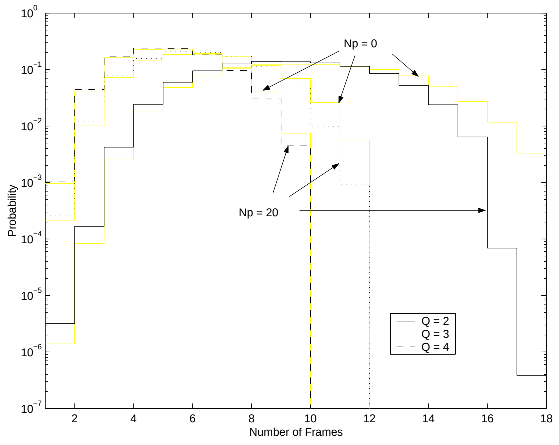


Figure 8.12: *The impact of Q and N_p on the delay density function*

Secondly, Figures 8.11 and 8.12 show that the influence of the polling threshold N_p decreases as the splitting factor Q increases (mean and quantiles); thus, the polling feature is the most attractive for the binary ISAP protocol. A general remark on N_p is that different values of N_p only result in a different behavior when there is at least one multiple of Q^2 in between.

Figures 8.13 and 8.14 demonstrate the influence of Q on the throughput results for $N_p = 0$ and $N_p = 20$. For $N_p = 0$ the highest throughput is obtained with the ternary scheme, except for very low load conditions where the binary scheme is slightly superior. For $N_p = 20$ we have the best results for the ternary scheme, in this case the binary scheme no longer dominates the quaternary scheme for λ around 1. Taking both the delay and throughput into account, we may conclude that it is better to use a ternary scheme as opposed to a binary one. The choice between the ternary and the quaternary is a tradeoff between the delay and throughput. It has to be mentioned that there do exist some values for N_p for which the binary scheme has better throughput results than the ternary scheme, e.g., for $27 = 3^3 \leq N_p < 32 = 2^5$.

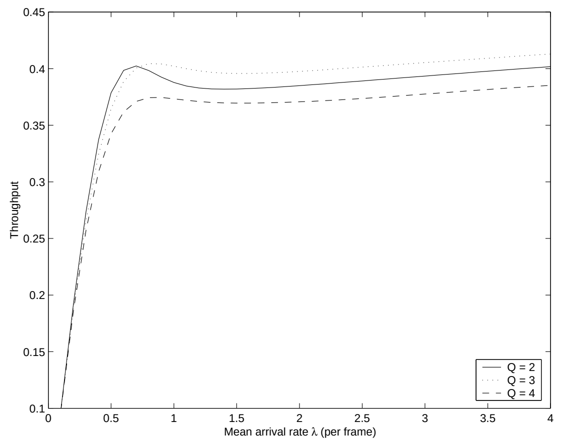


Figure 8.13: *The impact of* Q *on the throughput results for* $N_p = 0$

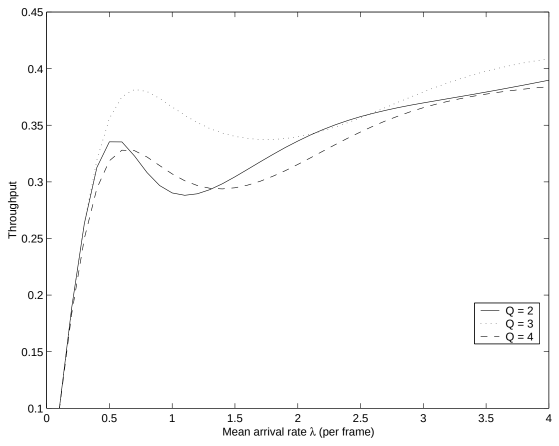


Figure 8.14: *The impact of* Q *on the throughput results for* $N_p = 20$

8.2.2 The Interaction between the Splitting Factor and the Starting Level

We only show results for $S_l = 0$ and $S_l = 1$, although the analytical model imposes no restraints on the value of the starting level S_l (expect that Q^{S_l} is bounded by L). Figures 8.15 and 8.16 show the influence of the starting level S_l and its interaction with Q for $N_p = 0$. First, the absolute delay improvement that we obtain for $S_l = 1$ is very similar in all three cases (binary, ternary and quaternary). In general, the absolute delay improvement that we obtain from a higher starting level is, to a certain extent, independent of the splitting factor Q.

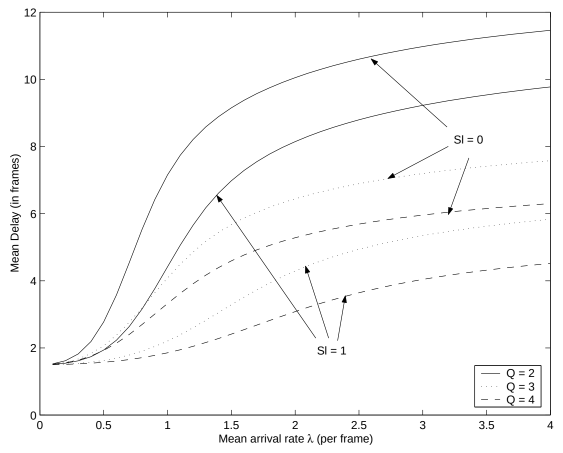


Figure 8.15: *The impact of* S_l *and* Q *on the mean delay*

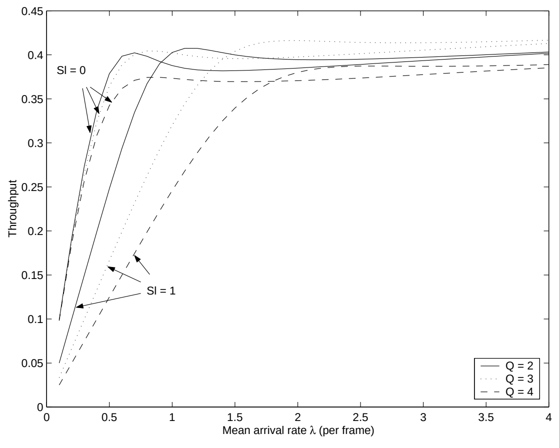


Figure 8.16: *The impact of* S_l *and* Q *on the throughput results*

As for the throughput, we always get a slight improvement when we increase the starting level to one, except under low load conditions. Also, the throughput losses suffered under low load conditions become more severe as Q increases. Therefore, if we want to combine a higher starting level ($S_l \geq 1$) with a higher splitting factor Q, we suggest that it is best to make the starting level S_l dynamic between S_{min} and S_{max}, with $S_{min} = 0$ or 1 (see Section 6.4 for the details).

8.3 Results for the M-ISAP Scheme

In this section we study the impact of the optional M_p parameter on the delay and throughput, by making use of the analytical model presented in Section 7.4. We restrict ourselves to the following scenario: $Q = 2$, $n = 8$ and $N_p = 32$. The optional M_p parameter is varied from 0 to 8. Notice, the behavior of M-ISAP and ISAP is identical if $M_p \leq 4$ (in general: $M_p \leq n - \lfloor \log_Q N_p \rfloor + 1$) and the behavior M-ISAP and ISA is identical if $M_p = 8$ (in general: $M_p = n$).

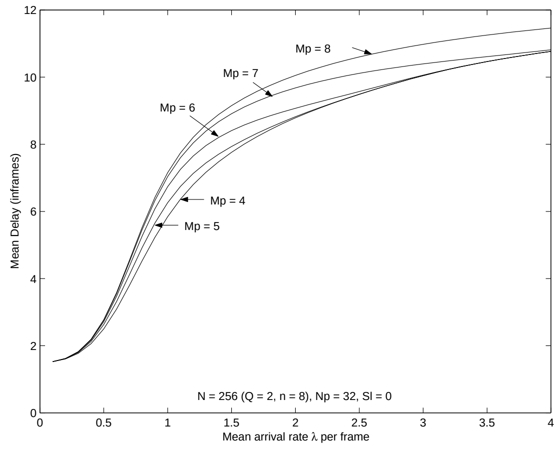


Figure 8.17: *The impact of M_p on the delay*

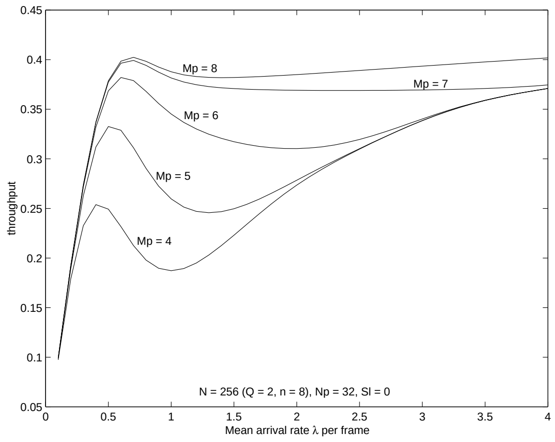


Figure 8.18: *The impact of M_p on the throughput*

Figures 8.17 and 8.18 demonstrate the usefulness of the M_p parameter: increasing M_p reduces the throughput losses caused by the polling feature, but increases the mean waiting time. The interesting part about M_p is that the throughput gains, for $M_p = 5$ and 6 (in general: $M_p = f(N_p)+1$ and $f(N_p)+2$, where $f(N_p) = n - \lfloor \log_Q N_p \rfloor + 1$), are much more significant than the delay losses. For instance, for $\lambda = 0.75$ we get an 8% throughput gain when increasing M_p from 4 to 5, while the mean delay increment is practically zero.

8.4 The Influence of L on ISAP

In this section we investigate the influence of the L parameter, i.e., the maximum number of contention slots allowed in one frame (see Section 6.1). The results presented were obtained using the package Mathematica and are therefore exact. As indicated in Section 7.5, we restrict ourselves to the binary case, although the analytical model can easily be generalized to capture splitting factors $Q > 2$. We consider MAC addresses with $n = 7$ bits, although $n = 8 - 10$ bits might be somewhat more realistic. The number of participants (MSs) in the CC therefore varies from $k = 2$ to 128 (sometimes we only show the results for $k \leq 60$ because no significant differences were observed for $k \geq 60$). The number of contention slots allowed in one frame equals $L = 4s$, with $4 \leq s \leq 16$ or $L = 128$. The trigger value N_p is also a multiple of 4 between 16 and 64.

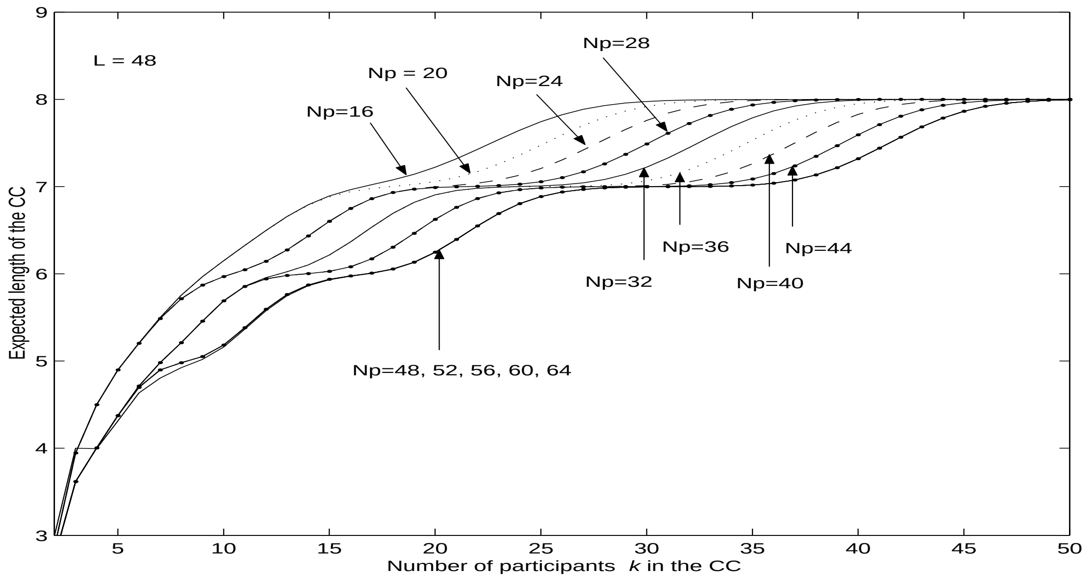


Figure 8.19: *The interaction between L and N_p with $L = 48$*

8.4.1 Tuning the Trigger Value N_p

In this section we focus on the interaction between the polling threshold N_p and the L parameter. Figure 8.19 ($L = 48$) shows that the expected length of the CC (in frames) decreases as N_p increases for $N_p \leq 48$. Indeed as long as $N_p \leq L$ polling only lasts one frame and therefore it always results in a delay improvement. More surprisingly, all the curves are almost identical when $N_p = 48, 52, 56$ and 60. To understand this let us compare the cases $N_p = 48$ and $N_p = 52$. Both these cases behave identical except when, at some level $i < 6$, the size of the remaining address space Y is larger than 48, but smaller than (or equal to) 52. In such a case we switch to polling if $N_p = 52$, namely, Y contention slots are included in the next two frames. Thus, the remaining length of the CC is two frames. When $N_p = 48$ it is very likely that the remaining length of the CC is also two frames. Indeed, the first frame to come contains level $i + 1$ of the tree (only one frame is required to support level $i + 1$ as $L = 48$ and $i < 6$) and the second frame to come is most likely used to poll the remaining contenders after level $i + 1$ (as it is highly probable that the size of the remaining address space will drop below 48). Finally, increasing N_p even more ($N_p = 64$) results in a somewhat larger delay for small values of k. Therefore, choosing $N_p > L$ might not be that useful.

Figure 8.20 shows the results for $L = 16$. It confirms that there is no use in choosing a polling threshold $N_p > L$ when we look at the expected delay. Moreover, the difference between two values of N_p is only significant if there is a multiple of L in between.

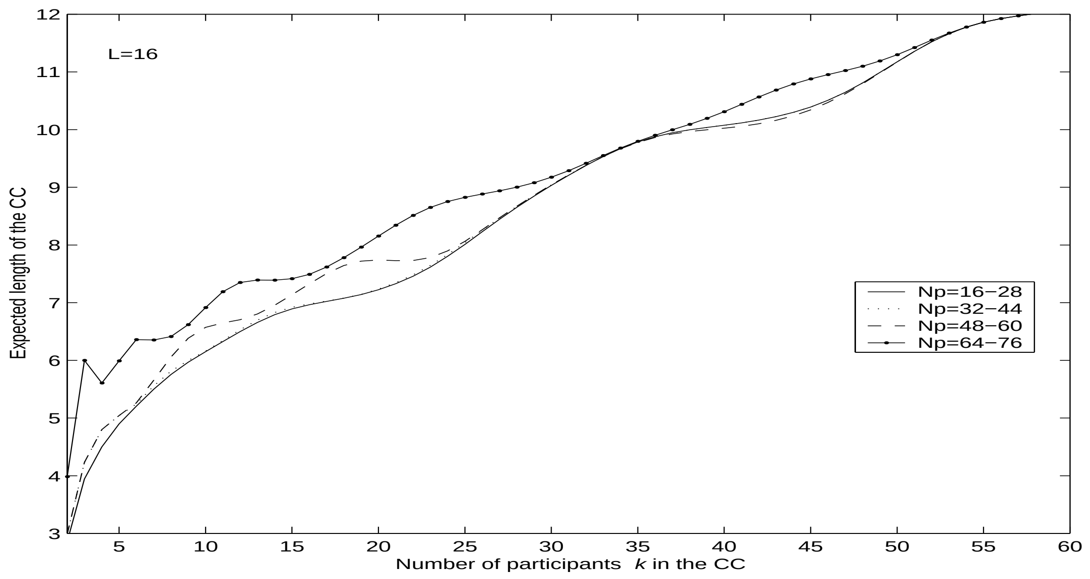


Figure 8.20: *The interaction between L and N_p with $L = 16$*

In general, with respect to the expected delay of the scheme, we conclude that the optimal choice for N_p is L. There is one exception to this rule: setting $N_p = 2^n$ with n small, e.g., $n < 8$, might result in a better delay: especially if k becomes large—that is, if the contention channel is highly loaded. For example, in a system with $N_p = 128$ and $L = 16$ (as shown in Figure 8.20) the length of the CC would be fixed and equal to 8 frames. The main disadvantage of choosing $N_p = 2^n$ is the low throughput that is obtained, leaving less slots available for contention free transmissions (see Section 6.1).

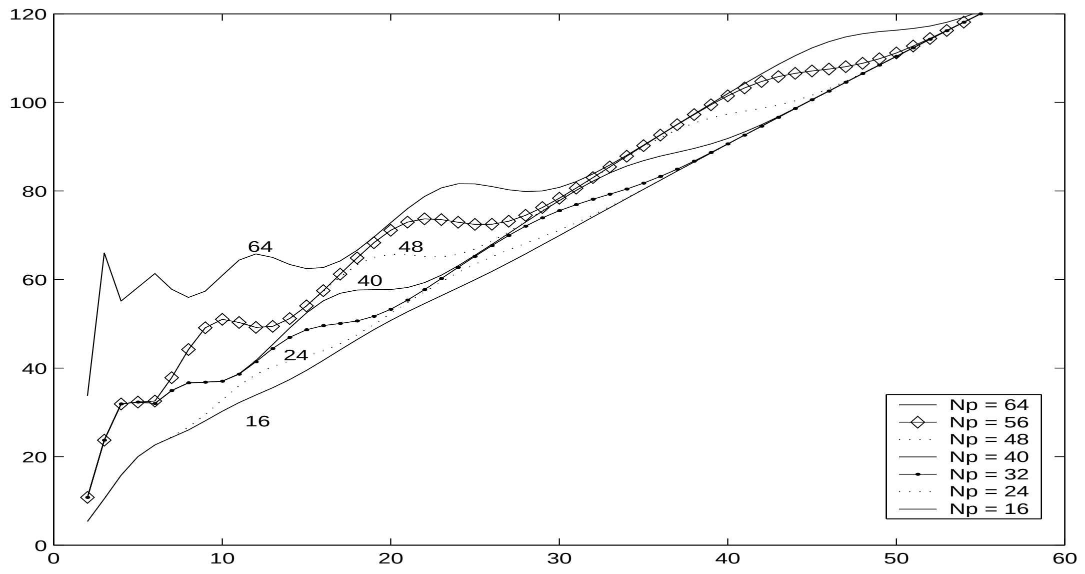


Figure 8.21: *$E[S]$ for different values of N_p*

The throughput of a CC with k participants can be defined as $k/E[S \mid X = k]$. Figure 8.21 shows that the expected number of slots in a CC (given that we have k contenders) always increases when N_p increases. Moreover, as N_p approaches zero, $E[S \mid X = k]$ approaches (decreases to) a linear curve for k large. Combining Figures 8.19, 8.20 and 8.21 we may conclude that N_p should always be chosen smaller than or equal to L. The closer we choose N_p to L the better the mean delay but the worse the throughput becomes.

8.4.2 The Influence of the Parameter L

In this section we investigate the influence of the maximum number of contention slots L allowed in one frame on the delay and throughput measures defined in Section 7.5. Figure 8.22 shows $E[F]$ for different values of $L \geq N_p = 16$. A number of conclusions can be drawn from this figure. Clearly, the less contention slots we allow in one frame the larger the delay becomes. Moreover, the delay improvements that we get when we increase L are the most significant if there is a power of 2 in between. In Section 8.4.1 we saw that different choices for N_p ($\geq L$) only resulted in a significant difference if there is a multiple of L in between. Because $N_p = 16$, a small power of two, it is tempting to believe that the difference between two choices of L is the most significant if there is a multiple of N_p in between. Numerical experiments have shown that this is generally not the case. Moreover, even if there is no power of two in between different choices of L, we still get a relevant impact on the mean delay.

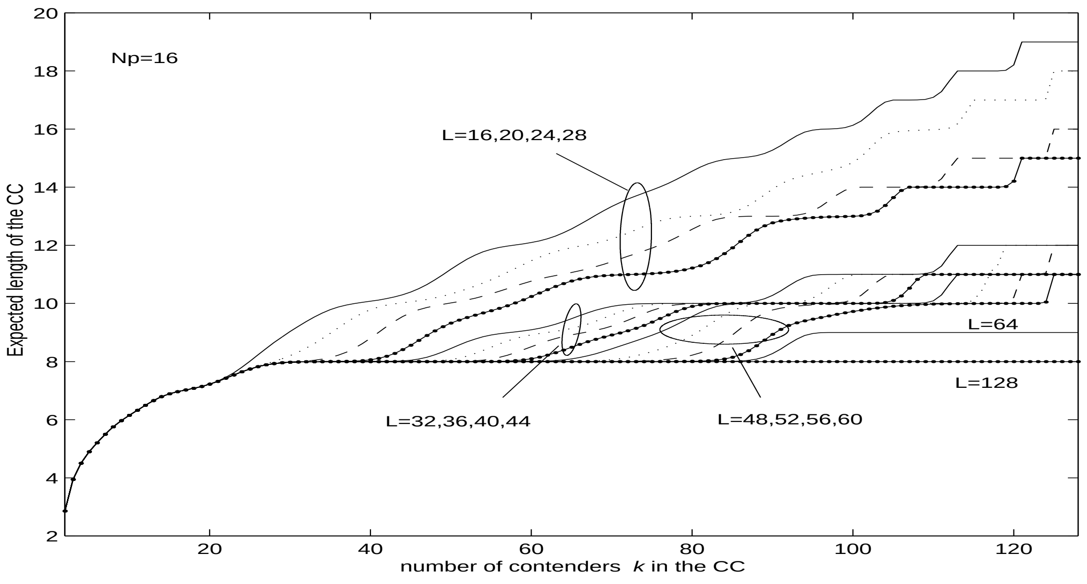


Figure 8.22: $E[F]$ *for different values of L*

Different values L_1 and L_2 (both bigger than N_p) do result in identical results when the number of contenders k is smaller than $\min(L_1, L_2)$; this follows from the fact that any level that is part of a CC with k contenders never requires more than k slots (in the Q-ary case: $kQ/2$ slots). Thus, even if we do not take L into account we can still get good approximate results for low and medium load situations, validating our approach presented in Sections 7.2 to 7.4.

Although we already demonstated that there is little use in choosing $N_p > L$, we also

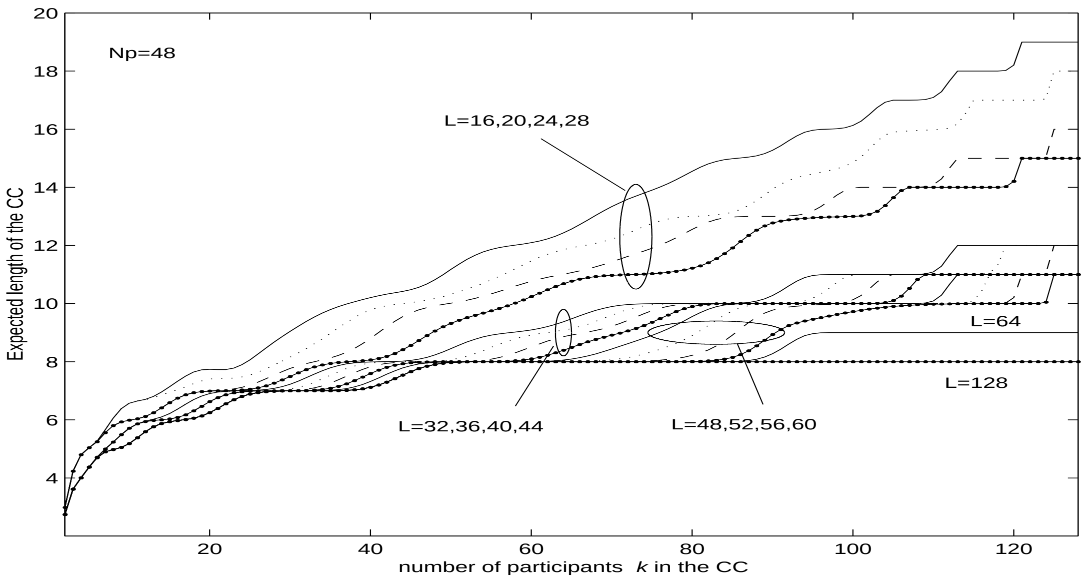


Figure 8.23: *$E[F]$ for different values of L*

include Figure 8.23 for reasons of completeness. The main purpose of Figure 8.23 is to demonstrate that different values L_1 and L_2 do not coincide for k smaller than $\min(L_1, L_2)$ when $N_p > \min(L_1, L_2)$.

8.4.3 Selecting the Starting Level S_l

In this section we investigate the interaction between the starting level S_l and the L parameter. In Figure 8.24 the influence of the starting level S_l on $E[F]$ is shown ($L = 16$

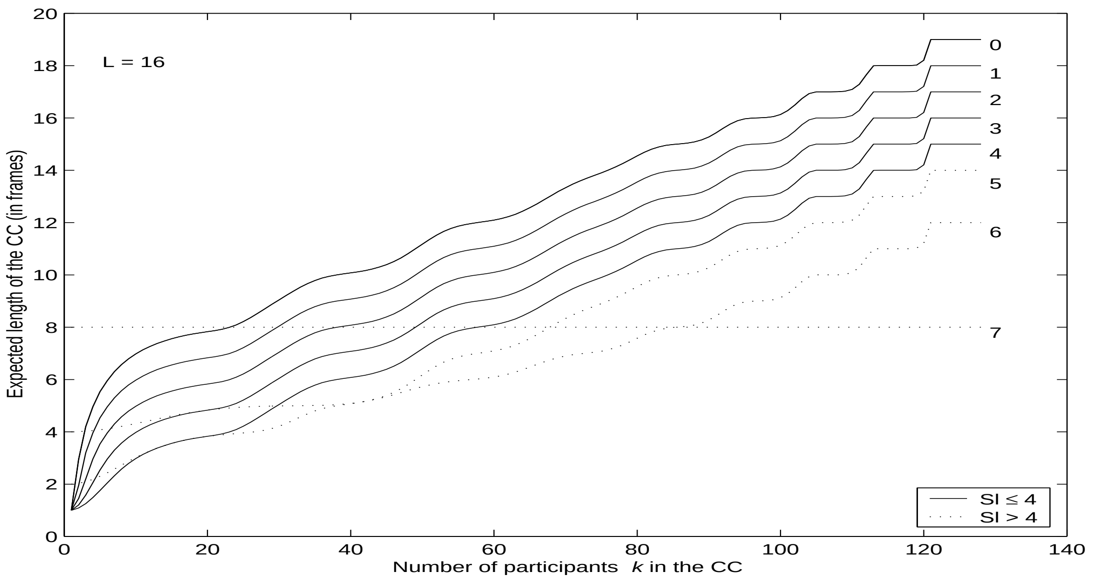


Figure 8.24: *$E[F]$ for different values of S_l*

and $N_p = 0$). For $S_l \leq 4$ the delay decreases for all values of k when increasing the starting level S_l. Moreover, the improvement that we get by increasing S_l by one is close to one frame. For $S_l > 4$ we still have a delay improvement for large values of k (a more significant one compared to $S_l \leq 4$), but a price is paid for smaller values of k. Note that for $S_l = 7$ we obtain a pure polling scheme. In general, looking from the delay perspective, we get the best results with $S_l = \log_2(L)$ if the contention channel has a low to medium load. For high loads a larger value for S_l might be considered.

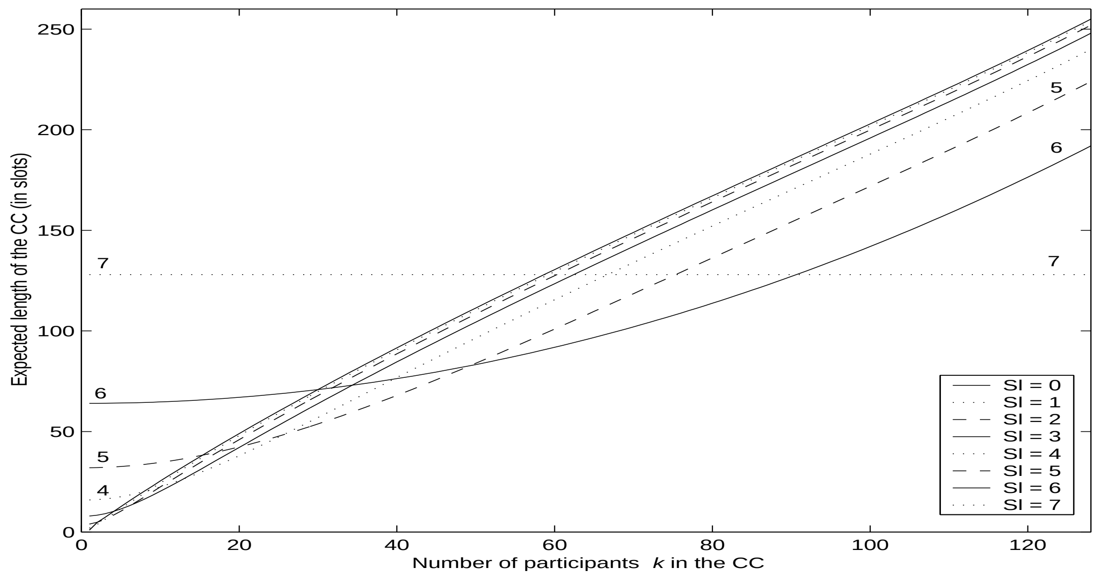


Figure 8.25: *$E[S]$ for different values of S_l*

In Figure 8.25 the throughput results are shown for different values of S_l. Notice that these results are independent of L. It demonstrates that increasing S_l, when the contention channel carries a low or medium load, increases the number of slots a CC requires. On the other hand if the load is high, better results are obtained for high values of S_l.

8.4.4 Stability Issues

In this section we investigate the influence of the protocol parameter L and the trigger value N_p on the stability of the scheme under Poissonian input traffic. We define the drift $D[k]$ of the protocol as $\min(2^n, \lambda E[F \mid X = k]) - k$, where λ is the expected number of arrivals per frame. A positive $D[k]$ implies that a CC with k contenders is generally followed by a CC with more contenders, a negative value indicates an expected decrease in the number of contenders in the CC. Finally, when the number of contenders k is such that $D[k] = 0$ the number of contenders is expected to remain the same, therefore we refer to these points as stability points. The scheme is expected to operate around these stability points for the majority of time.

Figure 8.26 shows the drift for $\lambda = 2, 3.5, 5, 6.5, 8$ and 9.5, with $L = 32$ and $N_p = 16$ or 32. With the exception of $\lambda = 9.5$ all the curves have a unique stability point. The curve with $\lambda = 9.5$ was included on purpose to show that in some rare cases the unique

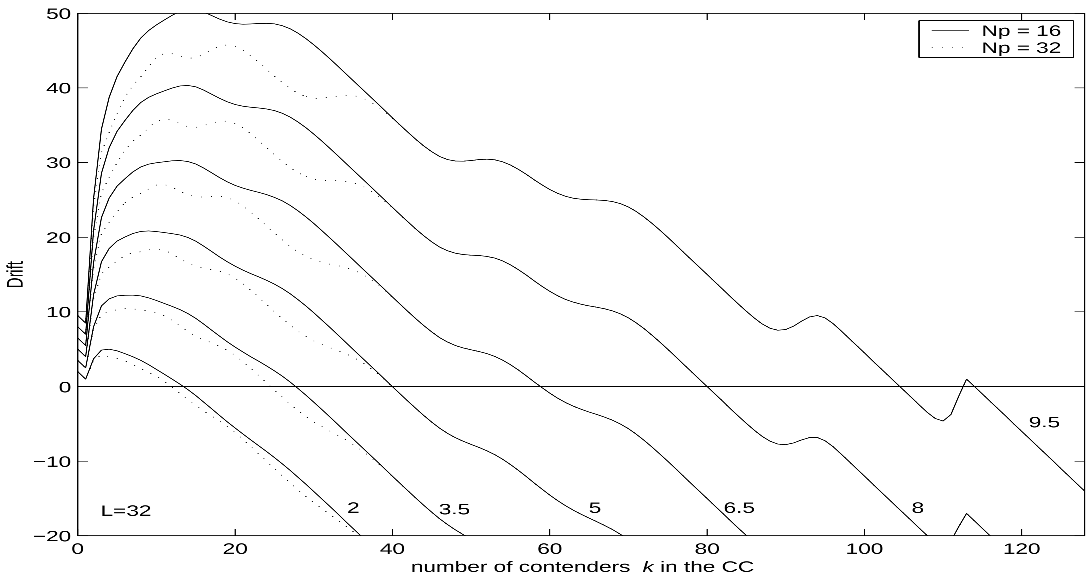


Figure 8.26: *Stability points for Poissonian input traffic*

stability point might split into two hardly separated stability points (this is due to the oscillations in the $E[F]$ curves). Nevertheless, these split stability points are not expected to endanger the general stability of the protocol. Comparing the results for $N_p = 16$ and $N_p = 32$, we see that the stability point of the protocol remains the same for $\lambda \geq 5$, as opposed to $\lambda < 5$ where we get a smaller stability point for $N_p = 32$. Thus, the delay improvement that we get by increasing $N_p(\leq L)$ is the most significant for systems with low to medium loads.

In conclusion, it should be clear that the introduction of L does not affect the stability of the protocol, though numerical experiments did show that the stability points might shift somewhat to the right when we decrease L in systems with a high load.

8.5 Conclusions

In this section we summarize the main conclusions drawn from the numerical examples presented in Sections 8.1 to 8.4. We discuss one parameter at a time starting with L, followed by Q, N_p, M_p and S_l. Although we restricted ourselves to $Q = 2$, when studying the impact of L, we intuitively generalize these conclusions to the Q-ary case.

The impact of L, the maximum number of contention slots allowed in a single frame, can be summarized as follows:

- Obviously, incrementing L reduces the delay suffered by a request.
- An increment of L from l_1 to l_2 is the most significant if one or more powers of Q are located within the interval $]l_1, l_2]$. Although, other increments are also useful.

- Under low or medium load conditions the influence of L is minor when chosen large enough.

As for the splitting factor Q, we have:

- Increasing the splitting factor Q results in smaller delays (mean and quantiles).
- From the throughput perspective we obtain, for most scenarios, the best results for the ternary scheme.
- A ternary scheme should be preferred above a binary one. The choice between the ternary and the quaternary is a tradeoff between the delay and throughput.

The influence of the polling threshold, N_p, can be summarized as follows:

- The polling threshold N_p should not be chosen larger than L, the maximum number of contention slots allowed in one frame.
- When selecting an appropriate value for N_p a tradeoff has to be made between the delay and throughput characteristics where a better delay is obtained for larger values of N_p ($\leq L$).
- Switching to polling has a more significant impact for smaller splitting factors Q. The delay improvements for $Q > 3$ do not seem to pay off against the complexity introduced by the polling mechanism. Therefore, one should not implement it for $Q > 3$.

As for the optional M_p parameter:

- The optional M_p parameter, to be used in combination with N_p, is useful to make the delay vs. throughput tradeoff, when selecting N_p, more attractive.

A fixed or variable starting level S_l has the following influence:

- If the load of the contention channel is low (or medium) the starting level S_l should not be chosen larger than $\log_Q(L)$. For $S_l \leq \log_Q(L)$ we get a similar tradeoff as with the polling threshold N_p, i.e., the larger S_l the better the delay and the worse the throughput becomes.
- For highly loaded systems it might still be useful to select $S_l > \log_Q(L)$ as this might result in better delay and throughput characteristics.
- A higher starting level does however result in a serious throughput degradation if the channel is poorly loaded. This throughput loss can be avoided by making the starting level variable (see Section 6.4).

Finally, we also indicated that the ISAP protocol often has a single stability point and should operate around this point for the majority of time. Further optimizations can be made by implementing multiple instances of ISA.

Conclusion

This thesis focuses on the performance evaluation of a family of algorithms used to solve the so-called multiple access problem that occurs in communication networks whenever multiple sending and receiving nodes are all connected to the same, single, shared link. Protocols, or algorithms, designed to solve this problem are known as multiple access protocols. Within this thesis we have analyzed the performance of a specific class of multiple access protocols commonly known as tree algorithms and this both from a theoretical and a more practical point of view. The thesis is subdivided into two parts.

The first analyzes the maximum stable throughput of tree algorithms, often referred to as their efficiency, under a number of idealized conditions. These conditions are used as the standard model of a multiple access link within the IEEE Information Theory Society [8]; hence, the multiple access problem is viewed from a theoretical perspective. The main difference with all prior work is that we have significantly relaxed the assumptions made on the arrival process—an arrival process is a stochastic process that specifies how new packets are generated by the users (senders) connected to the shared link. Instead of Poisson arrivals we consider a rich class of tractable Markovian arrival processes, which lend themselves very well to modeling bursty arrival processes arising in computer and communication networks—namely, we consider discrete time batch Markovian arrival processes (D-BMAPs). Tree algorithms can be further categorized into three subclasses: the blocked access, free access and grouped access class. The methods used to analyze the first subclass—see Chapter 2—are fairly common and originated in the early 1980s [41]. To a certain extent the same can be said about the grouped access class (although some complications do arise, see Chapter 5). The free access class is by far the most difficult to analyze (given the current state of the art results) and requested a very different and new approach, Chapters 3 and 4 are devoted to them. The key result is to view a tree algorithm with free access as a tree structured quasi-birth-death (QBD) Markov chain, the theory of which was developed during the late 1990s, and to study the stability of the algorithm by means of the recurrence of the Markov chain. The main conclusion drawn from the first part of the thesis is that the good stability characteristics of tree algorithms under Poisson arrivals are maintained under this rich class of arrival processes, thereby further extending the established theoretical foundation of tree algorithms. More detailed conclusions and key results are found at the end of each chapter.

In the second part of the thesis, we study tree algorithms from a more practical perspective. Many access systems—for instance, wireless broadband systems, hybrid fiber coaxial (HFC) networks or passive optical networks (PONs)—have a point-to-multipoint architecture. The single end point, referred to as the access point (AP), operates as a centralized controller, that is, it decides which of the end nodes gets to transmit a packet to the AP. To make this decision, end nodes need to declare their bandwidth requirements to the access point (AP). This information is then used by the AP to schedule all uplink transmissions, that is, transmissions from an end node to the AP, according to the traffic characteristics and the quality of service (QoS) agreed upon. A problem of central importance is how the end nodes inform the AP about their bandwidth needs, a problem that has received considerable attention of the IEEE Communication Society. In the second part of this thesis, we address this problem in the context of wireless broadband access networks and we provide a detailed analysis of the *Identifier Splitting Algorithm combined with Polling (ISAP)* —see Chapter 6. The Identifier Splitting Algorithm is a tree algorithm that was introduced during the European RACE project 2067 on Mobile Broadband Systems (MBS). We have enhanced this algorithm with a polling mechanism and studied the influence of its parameters on the delay and throughput characteristics by means of several analytical models. These models, presented in Chapter 7, combine elementary probability theory, queueing theory, combinatorics and the theory of Markov chains. A summary of the main conclusions drawn from the numerical results, presented in Chapter 8, is given in Section 8.5.

Nederlandse Samenvatting

Deze thesis handelt over de performantie evaluatie van een verzameling algoritmen die gebruikt worden om het zogenaamde “multiple access” probleem—dat optreedt in communicatie netwerken telkemale meerdere zendende en ontvangende gebruikers gebruik maken van éénzelfde, gezamelijke communicatie link—op te lossen. Algoritmen, of protocols, die ontworpen zijn om aan dit probleem een antwoord te bieden, zijn gekend als “multiple access” algoritmen. Binnen het kader van deze thesis wordt de performantie van een welbepaalde klasse van multiple access algoritmen, genaamd *tree* algoritmen, geëvalueerd. Deze evaluatie gebeurt zowel vanuit een theoretisch oogpunt, alsook vanuit een meer praktische invalshoek. Vandaar dat de thesis ook is opgesplits in twee delen.

In het eerste deel wordt de maximale stabiele throughput, d.w.z., de maximale verwerkingscapaciteit of efficiëntie, bestudeerd, en dit onder een aantal geïdealizeerde condities. Deze condities worden, door de IEEE Information Theory Society, veelal gehanteerd als het standaard model voor multiple access communicatie links. Gegeven de ideologie die deze organizatie hanteert, kunnen we stellen dat het probleem bekeken wordt vanuit een meer theoretisch oogpunt. Het grote verschil met al het voorgaande werk bestaat erin dat we de veronderstellingen gemaakt op het aankomstenproces—het aankomstenproces is een stochastisch proces dat aangeeft wanneer de gebruikers nieuwe pakketten aanmaken—sterk versoepeld hebben. In plaats van Poisson aankomsten te veronderstellen, beschouwen we een erg rijke klasse van aankomstenprocessen, die uiterst geschikt is voor het modeleren van de meer onregelmatige aankomstpatronen die we terug vinden in moderne communicatie netwerken— namelijk, discrete tijds batch Markoviaanse aankomstenprocessen (D-BMAPs).

De beschouwde algoritmen, d.w.z. de *tree* algoritmen, kunnen verder ingedeeld worden in drie categorieën. De categorie waartoe een bepaald algoritme behoort, hangt af van de strategie dat het hanteert om nieuwe aankomsten in het schema te betrekken. Zo zijn er algoritmen met geblokkeerde, vrije en gegroepeerde toegang. De methode die gehanteerd werd voor de evaluatie van de eerste categorie van algoritmen—dat is, deze met geblokkeerde toegang, zie Hoofdstuk 2—is vrij gebruikelijk en werd reeds in het begin van de jaren tachtig ontwikkeld [41]. Tot op zeker hoogte kan hetzelfde gezegd worden omtrent de algoritmen met gegroepeerde access, zij het dat er toch een aantal complicaties optreden, zie Hoofdstuk 5. De categorie met de vrije toegang is veruit de moeilijkste om te evalueren, gegeven de huidige stand van zaken, vandaar dat deze ook vroeg om een geheel nieuwe benadering. Hoofdstuk 3 en 4 zijn hieraan gewijd. Het belangrijkste resultaat bestaat erin om deze algoritmen te zien als een boomgestructureerde QBD (“Quasi-Birth-Death”)

Markov keten, een theorie die zelf pas op het einde van de jaren negentig ontwikkeld is. De hoofdconclusie van het eerste deel van de thesis is dat de goede stabiliteitskenmerken, in het geval van Poisson aankomsten, bewaard blijven wanneer we D-BMAP aankomsten beschouwen. Dit resultaat draagt dus erg bij tot de verdere theoretische onderbouw van *tree* algoritmen als oplossing voor het multiple access probleem. De overige conclusies worden samengevat op het einde van elk hoofdstuk.

In het tweede deel van de thesis worden de *tree* algoritmen vanuit een meer praktische invalshoek bekeken. Vele access netwerken—bijvoorbeeld, draadloze netwerken, HFC ("Hybrid Fiber Coaxial") netwerken en PON ("Passive Optical Networks") netwerken—hebben een gecentralizeerde architectuur. Concreet betekent dit dat al het verkeer van of naar het netwerk loopt via een enkel knooppunt, dat we het access punt (AP) noemen. Het AP bepaalt ook, en dit op elk ogenblik, welke eindgebruiker informatie mag versturen naar het AP (en dus naar het netwerk toe). Om deze beslissing te kunnen nemen, moet elk van de eindgebruikers zijn huidige behoefte aan bandbreedte kenbaar maken aan het AP. Het AP zal dan een beslissing maken op basis van de verkregen informatie en dit in overeenkomst met het contract dat bestaat tussen de eindgebruiker en de service provider (die eigenaar is van de netwerk infrastructuur). Een belangrijke vraag hierbij is: Hoe kan een eindgebruiker zijn huidige behoefte aan bandbreedte kenbaar maken aan het AP ? Dit probleem heeft al heel wat aandacht gekregen van de IEEE Communication Society. In het tweede deel van deze thesis bekijken we dit probleem in het licht van draadloze breedband access netwerken en maken we een uitgebreidde analyse van het *Identifier Splitting Algoritme in combinatie met Polling (ISAP)*—zie Hoofdstuk 6.

Het *Identifier Splitting Algoritme* is een *tree* algoritme dat voor het eerste geïntroduceerd werd tijdens het Europeese RACE 2067 project dat handelt over mobiele breedband systemen (MBS). In het kader van deze thesis hebben we dit algoritme verrijkt met een polling mechanisme en hebben we vervolgens, op basis van een aantal analytische modellen, de invloed op de performantie—dat is, de wachttijd en de efficiëntie—van de verschillende parameters van het algoritme bestudeerd. Deze analytische modellen worden voorgesteld in Hoofdstuk 7 en maken gebruik van elementaire kanstheorie, queueing theorie, combinatoriek en Markov ketens. Een samenvatting van de belangrijkste conclusies wordt gegeven op het einde van Hoofdstuk 8.

Bibliography

[1] N. Abramson. The ALOHA system - another alternative for computer communications. In *Proc. Fall Joint Comput. Conf., AFIPS Conf.*, page 37, 1970.

[2] D. Aldous. Ultimate instability of exponential back-off protocol for acknowledgement-based transmission control of random access communication channels. *IEEE Transactions on Information Theory*, IT-33:219–223, 1987.

[3] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall Int., Inc., 1992.

[4] C. Blondia. A discrete-time batch markovian arrival process as B-ISDN traffic model. *Belgian Journal of Operations Research, Statistics and Computer Science*, 32(3,4), 1993.

[5] C. Blondia and O. Casals. Statistical multiplexing of VBR sources: A matrix-analytical approach. *Performance Evaluation*, 16:5–20, 1992.

[6] C. Blondia and F. Geerts. The correlation structure of the output of an ATM multiplexer. In *Fifth IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, pages 235–250. Kluwer Academic Publ., 1999.

[7] J.I. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Trans. Inform. Theory*, 25(5):319–329, 1979.

[8] I. Cidon and M. Sidi. Conflict multiplicity estimation and batch resolution algorithms. *IEEE Trans. on Information Theory*, IT-34(1):101–110, Jan 1988.

[9] D. Vázquez Cortizo, J. García, and C. Blondia. FS-ALOHA++, a collision resolution algorithm with QOS support for the contention channel in multiservice wireless LANs. In *Proc. of IEEE Globecom*, Dec 1999.

[10] D. Vázquez Cortizo, J. García, C. Blondia, and B. Van Houdt. FIFO by sets ALOHA (FS-ALOHA): a collision resolution algorithm for the contention channel in wireless ATM systems. *Performance Evaluation*, 36-37:401–427, 1999.

[11] T. Daniëls. *Asymptotic Behaviour of Queueing Systems.* PhD thesis, University of Antwerp (UA), 1999.

[12] D.Z. Du and F.K. Hwang. Combinatorial group testing and its applications. *World Scientific*, 1993.

[13] G. Fayolle, P. Flajolet, M. Hofri, and P. Jacquet. Analysis of a stack algorithm for random multiple-access communication. *IEEE Transactions on Information Theory*, IT-31(2):244–254, 1985.

[14] G. Fayolle and M. Hofri. On the capacity of a collision channel under stack-based collision resolution algorithms. Technical report, 1983.

[15] C. Fernàndez and S. Sallent. Evaluation of allocation policies on hybrid fiber-coax broadband access networks for contention-type traffic. In *Proc. of Broadband Communications (BC)*, Hong Kong, 1999.

[16] P. Flajolet. Evaluation de protocols de communication: aspects mathematics. Technical Report 797, INRIA, 1988.

[17] P. Flajolet and P. Jacquet. Analytic models for tree communication protocols. Technical Report 648, INRIA, 1987.

[18] H.R. Gail, S.L. Hantler, and B.A. Taylor. Non-sik-free M/G/1 and G/M/1 type markov chains. *Adv. Appl. Prob.*, 27:733–758, 1997.

[19] L.A. Goldberg, M. Jerrum, S. Kannan, and M. Paterson. A bound on the capacity of backoff and acknowledgement-based protocols. Technical Report 365, Dept. of Computer Science, University of Warwick, Coventry CV4 7 AL, UK, Jan 2000.

[20] N. Golmie, F. Mouveaux, and D. Su. A comparison of mac protocols for hybric fiber/coax networks: Ieee 802.14 vs. mcns. In *Proc. of the 16th Int. Conf. on Comm.*, pages 266–272, Vancouver, Canada, June 1999.

[21] N. Golmie, Y. Saintillan, and D.H. Su. A review of contention resolution algorithms for IEEE 802.14 networks. *IEEE Communication Surveys*, 2(1), 1999.

[22] A.G. Greenberg, P. Flajolet, and R.E. Ladner. Estimating the multiplicity of conflicts to speed their resolution in multiple access channels. *Journal of the Association of Computing Machinery*, 34(2):289–325, 1987.

[23] D. Gross and C.M. Harris. *Fundamentals of Queueing Theory.* John Wiley and Sons, New York, 1974.

[24] Q. HE. Classification of markov processes of M/G/1 type with a tree structure and its applications to queueing models. *O.R Letters*, 26:67–80, 1999.

[25] Q. HE. Classification of markov processes of matrix M/G/1 type with a tree structure and its applications to the MMAP[K]/G[K]/1 queue. *Stochastic Models*, 16(5), 2000.

[26] Q. He and A.S. Alfa. The discrete time MMAP[K]/PH[K]/1/LCFS-GPR queue and its variants. In *Proc. of the 3rd Int. Conf. on Matrix Analytic Methods*, pages 167–190, Leuven (Belgium), 2000.

[27] Q. He and M.F. Neuts. Markov chains with marked transitions. *Stochastic Processes and their Applications*, 74:37–52, 1998.

[28] P.A. Humblet. On the throughput of channel access algorithms with limited sensing. *IEEE Trans. on Comm.*, COM-34:345–347, April 1986.

[29] N.L. Johnson and S. Kotz. *Urn models and their applications.* John Wiley and sons, Inc., New York, 1977.

[30] F.P. Kelly and I.M. MacPhee. The number of packets transmitted by collision detect random access schemes. *Annals of Probability*, 15:1557–1568, 1987.

[31] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Litton Educational Publishing, Inc., New York, 1960.

[32] L. Kleinrock. *Queueing Systems Vol. II.* Wiley, New York, 1976.

[33] J.F. Kurose and K.W. Ross. *Computer Networking: A top-down approach featuring the internet.* Addison and Wesley, New York, 2001.

[34] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods and stochastic modeling.* SIAM, Philadelphia, 1999.

[35] Y-D. Lin. On IEEE 802.14 medium access control protocol. *IEEE Communication Surveys*, 1(1), 1998.

[36] Y-D Lin, W-M Yin, and C-Y Huang. An investigation into HFC MAC protocols: mechanisms, implementation, and research issues. *IEEE Communication Surveys*, 3(3), 2000.

[37] M. Listanti, F. Mascitelli, and A. Mobilia. D2MA: A distributed access protocol for wireless ATM networks. In *Proc. IEEE Infocom*, 1997.

[38] U. Loher. *Information-theoretic and genie-aided analyses of random-access algorithms.* PhD thesis, Swiss Federal Institute of Technology, Zurich, 1998. DISS ETH No. 12627.

[39] D.M. Lucantoni. New results on the single server queue with a batch markovian arrival process. *Stochastic Models*, 7(1):1–46, 1991.

[40] D.M. Lucantoni, K.S. Meier-Hellstern, and M.F. Neuts. A single server queue with server vacations and a class of non-renewal arrival processes. *Adv. in Applied Probability*, 22:676–705, 1990.

[41] J.L. Massey. Collision resolution algorithms and random-access communication. In G. Longo, editor, *Multi-Users Communication Networks*, pages 73–137, Wien-New York, 1981. CISM Courses and Lectures No. 256, Springer Verlag.

[42] J.L. Massey. Some new approaches to random-access communications. In P.J. Courtois and G. Latouche, editors, *Performance '87.* Elsevier Science Publishers B.V. (North Holland), 1987.

[43] P. Mathys and P. Flajolet. Q-ary collision resolution algorithms in random-access systems with free or blocked channel access. *IEEE Transactions on Information Theory*, IT-31(2):217–243, 1985.

[44] MBS project (RACE II). http://comnets.rwth.aachen.de/project/mbs/home.html.

[45] R.M. Metcalfe and D. Boggs. Ethernet: Distributed packet switching for local computer networks. *Comm. ACM*, 19:217–243, 1976.

[46] M.F. Neuts. Markov chains with applications in queueing theory, which have a matrix geometric invariant probability vector. *Adv. Appl. Prob.*, 10:185–212, 1978.

[47] M.F. Neuts. *Matrix-Geometric Solutions in Stochastic Models, An Algorithmic Approach.* John Hopkins University Press, 1981.

[48] M.F. Neuts. *Structured Stochastic Matrices of M/G/1 type and their applications.* Marcel Dekker, Inc., New York and Basel, 1989.

[49] K. Pahlavan and A.H. Levesque. *Wireless Information Networks.* John Wiley and Sons, Inc., New York, 1995.

[50] D. Petras. Medium access control protocol for wireless, transparent ATM access in MBS. RACE Mobile Telecommunications Summit, (Cascais, Portugal), 1995.

[51] D. Petras and A. Krämling. Fast collision resolution in wireless ATM networks. In *2nd Mathmod*, Vienna, Austria, Feb 1997.

[52] D. Petras and A. Krämling. Wireless ATM: Performance evaluation of a DSA++ MAC protocol with fast collision resolution by a probing algorithm. *Int. Journal of Wireless Information Networks*, 4(4):215–224, 1997.

[53] Broadband optical access systems based on passive optical networks (PON). Recommendation G.983.1, ITU-T, 1998.

[54] M. Prögler, R. Dinis, N. Esseling, A. Gusmão, and A. Krämling. Specification of the air inferface. Deliverable DE04, 1997.

[55] V. Ramaswami. Nonlinear matrix equations in applied probability - solution techniques and open problems. *SIAM review*, 30(2):256–263, June 1988.

[56] V. Ramaswami. The generality of QBD processes. In *Proc. of the 2nd Int. Conference on Matrix Analytical Methods*, Winnipeg, Manitoba, 1998.

[57] SAMBA project (ACTS). http://hostria.cet.pt/samba/index.htm.

[58] E. Seneta. *Non-negative Matrices and Markov Chains.* Springer-Verlang, New York, 1981.

[59] K.M. Sivalingam, M.B. Srivastava, P. Agrawal, and J.C. Chen. Low-power access protocols based on scheduling for wireless and mobile ATM networks. In *Proc. IEEE Int. Conf. on Universal Personal Communications (ICUPC)*, pages 429–433, San Diego, CA, Oct. 1997.

[60] K. Spaey and C. Blondia. Circulant matching method for multiplexing ATM traffic applied to video sources. In *Proc. Perf. of Infor. and Comm. Systems (PICS)*, pages 234–245, Lund (Sweden), 1998.

[61] K. Sriram and W. Whitt. Characterizing superposition arrival processes in packet multiplexers for voice and data. *IEEE JSAC*, SAC-4(6):833–846, Sept. 1986.

[62] T. Takine, B. Sengupta, and R.W. Yeung. A generalization of the matrix M/G/1 paradigm for markov chains with a tree structure. *Stochastic Models*, 11(3):411–421, 1995.

[63] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall Int., Inc., 1989.

[64] B. S. Tsybakov and V.A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Inform*, 14(4):32–59, 1978.

[65] B. Van Houdt and C. Blondia. Analysis of a Q-ary identifier splitting algorithm combined with polling for contention resolution in a wireless ATM access network. In *Proc. of the ITC Spec. Seminar on Mobile Systems and Mobility*, Lillehammer (Norway), 2000.

[66] B. Van Houdt and C. Blondia. Analysis of an identifier splitting algorithm combined with polling for contention resolution in a wireless ATM access network. *IEEE J. on Selected Areas in Comm.*, 18(11):2345–2355, 2000.

[67] B. Van Houdt and C. Blondia. Performance analysis of the identifier splitting algorithm with polling in wireless ATM networks. *Int. Journal of Wireless Information Networks*, 7(2):91–104, 2000.

[68] B. Van Houdt and C. Blondia. Stability and performance of stack algorithms for random access communication modeled as a tree structured QBD Markov chain. *To Appear in Stochastic Models*, 17(3), 2001.

[69] B. Van Houdt and C. Blondia. Throughput of q-ary splitting algorithms for contention resolution in communication networks. *To appear in Adv. in Performance Analysis*, 2001.

[70] B Van Houdt, C. Blondia, and O. Casals. Packet level performance characteristics of a mac protocol for wireless ATM LANs. In *Proc. of the 24th Conf. on Local Comp. Netw. (LCN)*, Lowell Massachusetts, 1999.

[71] B. Van Houdt, C. Blondia, O. Casals, and J. García. Performance evaluation of a MAC protocol for wireless ATM networks supporting the ATM service categories. In *Proc. of 2nd ACM Int. Workshop on Wireless Mobile Multimedia (WoWMoM)*, Seattle, 1999.

[72] B. Van Houdt, C. Blondia, O. Casals, J. García, and D. Vázquez Cortizo. A MAC protocol for wireless ATM systems, supporting the service categories. In *Proc. of the 16th ITC*, Edinburgh, UK, June 1999.

[73] R. von Mises. *Mathematical Theory of Probability and Statistics.* Academic Press Inc., New York, 1964.

[74] B. Walke, D. Petras, and D. Plassmann. Wireless ATM: Air interface and network protocols of the mobile broadband system. *IEEE Personal Communications Magazine*, 3(2):50, 1996.

[75] K. Wuyts and R.K. Boel. A matrix geometric algorithm for finite buffer systems with B-ISDN applications. In *Proc. ITC Specialist Seminar*, pages 265–276, Lund (Sweden), 1996.

[76] K. Wuyts, B. Van Houdt, R.K. Boel, and C. Blondia. Matrix geometric analysis of discrete time queues with batch arrivals and batch departures with applications to B-ISDN. In *Proc. of the 16-th ITC conference*, Edinburgh (UK), 1999.

[77] Q. Ye. High accuracy algorithms for solving nonlinear matrix equations in queueing models. In *Proc. of the 3rd Int. Conf. on Matrix Analytical Methods (MAM3)*, Leuven (Belgium), 2000.

[78] R.W. Yeung and A.S. Alfa. The quasi-birth-death type markov chain with a tree structure. *Stochastic Models*, 15(4):639–659, 1999.

[79] R.W. Yeung and B. Sengupta. Matrix product-form solutions for markov chains with a tree structure. *Adv. Appl. Prob.*, 26:965–987, 1994.