Markov chains and decision processes for congestion avoidance and power control

Balakrishna Prabhu

Université de Nice - Sophia Antipolis – UFR Sciences
École Doctorale STIC

THÈSE

Présentée pour obtenir le titre de :
Docteur en Sciences de l'Université de Nice - Sophia Antipolis

Spécialité : Informatique

par

Balakrishna J. Prabhu

Thèse dirigée par M. Eitan ALTMAN et M. Konstantin AVRACHENKOV,
et préparée à l'INRIA Sophia Antipolis, projet MAESTRO

Markov chains and decision processes for congestion avoidance and power control

Soutenue publiquement à l'INRIA le 4 octobre 2004 à 10h30 devant le jury composé de :

Président :	Eitan	Altman	INRIA
Rapporteurs :	Ger	Koole	Vrije Universiteit
	Jorma	Virtamo	Helsinki University of Technology
Examinateurs :	Konstantin	Avrachenkov	INRIA
	Thomas	Bonald	France Télécom R&D et ENS
	Rayadurgam	Srikant	University of Illinois at Urbana-Champaign

THÈSE

Markov chains and decision processes for congestion avoidance and power control

Chaînes de Markov et processus de décision markoviens pour le contrôle de congestion et de puissance

Balakrishna J. Prabhu

Octobre 2005

Acknowledgments

It is a great pleasure to acknowledge the immense influence that Eitan Altman and Konstantin Avrachenkov, my research advisors, have had not only on the work presented in this thesis but also otherwise. I am grateful to them for having given me the opportunity to begin this work, the freedom to carry it out, and the support and encouragement to bring it to a conclusion. Their understanding of phenomena and alacrity is exemplary. Despite their success, their humility and willingness to lend an ear is inspirational.

I am grateful to Philippe Nain, the scientific leader of our team, for having hosted me as a part of his team during the course of this thesis.

A large part of this thesis owes its existence to the work done in collaboration with Arzad Alam Kherani. My sincere gratitude for all the delightful discussions.

I wish to thank Vivek Borkar for hosting me as summer intern at TIFR, Mumbai. It gave me the opportunity to learn new tools and methods, and to carry out a part of this work.

I owe a debt of gratitude to Rudesindo Núñez Queija for arranging an internship at CWI, Amsterdam, and for making it a great learning experience. I have greatly benefitted from the enriching discussions we have had during the past year and a half. Though the work has not yet reached a conclusion, the outcome of these discussions is partly reflected in this thesis.

It is a great privilege to have Ger Koole, Jorma Virtamo, Thomas Bonald, and R. Srikant as members of the thesis committee. I wish to thank them for having accepted this task. I thank especially Ger Koole and Jorma Virtamo, the thesis reviewers, for having carefully read the manuscript and for the suggestions and comments which have improved the presentation of the thesis.

I would like to thank Sara Alouf and Nicolas Bonneau who were kind enough to patiently read the French part of this thesis, and correct the numerous errors. Any errors that remain are entirely my responsibility.

The work was carried out in a lively and friendly atmosphere. I thank the members of Maestro/Mistral project – Ahmad Al-hanbali, Sara Alouf, Urtzi Ayesta, Nicolas Bonneau, Florence Clevenot, Rachid El-Azouzi, Robin Groenevelt, Daniele Miorandi – and Vijay Arya who contributed to this atmosphere. I am grateful to Ephie Deriche, our administrative assistant, who was always helpful with administrative details.

I am deeply grateful to INRIA, France Télécom, CEFIPRA, EURO-NGI, Project Van Gogh, and Project ECONET for having provided the financial support which made this work possible.

Last but not least, I would like thank the members of my family for their constant support and their presence.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Information highway, or the Internet, today connects millions of users across all the continents. As is true with any road network, an efficient mechanism to regulate the packet traffic is necessary in order to avoid congestion on the Internet. For example, a telephone network operator blocks incoming users when the network gets saturated. The architects of the Internet, on the other hand, designed a system which gave access to every incoming user. The design also had a fundamental difference from the conventional telephone network. Unlike the telephone network in which the operator controlled and regulated all the traffic, the architects of the Internet confided this task with the users. The successful deployment of this paradigm required every end user to be aware of the situation within the network and to react appropriately. For example, the existing users would have to reduce their transmission rates to allow an incoming user to transmit at a reasonable rate. The end users would, therefore, control and regulate their own traffic which traversed the Internet.

The transmission control protocol (TCP)[Jac88] was proposed and implemented as a distributed congestion control algorithm which regulated every user's own traffic so as to meet the design aim of fair and efficient sharing of the network resources. In addition to regulating the traffic, TCP also ensured that each packet was delivered eventually to its destination. The algorithm can be summarized as follows. The TCP algorithm probes the network by sending a small number of packets. If the packets are successfully received at the destination, an acknowledgement is sent by the destination to the sender. The acknowledgements indicate the availability of resources in the network. The sender reacts to the acknowledgements by sending larger number of packets. Eventually, the transmission rate exceeds the rate that the network can provide (i.e., congestion sets in), thereby resulting in a loss of packets. The sender reacts to the loss of packets by reducing the sending rate by half. The additive increase and multiplicative decrease (AIMD) algorithm just described proved to be fair and efficient in sharing the network capacity. The TCP algorithm is today used by a dominant proportion of Internet applications for reliable transfer of data.

Although TCP evolved along with the Internet, modifications were mainly restricted to ensure proper functioning of the algorithm in various environments (for example, on wireless links or to recover from multiple packet losses). The modifications were mainly engendered by the problems encountered by casual end users (i.e., users with low quantum of data transfers) of the Internet. The Internet was used not only by casual users but also by physics laboratories to exchange experimental data. The volume of data to be transferred was one or two orders of magnitude larger than what

typical user applications would transfer. These labs were connected by links of capacity larger than that available to a casual Internet user. During the course of such long file transfers over high speed links, the additive increase algorithm of TCP was identified as one of the causes of inefficient utilization of the large bandwidth. For example, suppose a session transferring 1000byte packets at 500Mbps over a link with round trip delay of 200ms encounters a packet loss due to imperfections in the optical fibre. The TCP algorithm would react to this loss by reducing its rate by half. It would now take close to fifty minutes before the transmission rate ramps up to the original 500Mbps. Since the loss is not due to congestion in the network, the transfer rate would be below the optimal rate for a relatively long duration. After detecting a packet loss, the time it takes for the additive increase algorithm to attain the available capacity increases linearly with the capacity. This linear dependence on the capacity was considered to be a drawback in the Internet where the link speeds were increasing rapidly every year.

The Internet community put forth several proposals to improve the performance of TCP over high speed links. These proposals focused on altering the additive increase algorithm to a more adaptive increase algorithm. A few of the proposals which have been discussed include FAST [JWL04], WestWood+ [GM04], HighSpeed TCP [Flo03], CuBIC [XHR04], and Scalable TCP [Kel03b]. We discuss briefly the salient features of these proposals. The TCP versions (presently deployed, and the new proposals) can be coarsely classified into two categories. The classification is based on the signals which are interpreted as congestion indications. The "loss-based" algorithms interpret only packet losses as signs of congestion. Examples of TCP versions in this category are Tahoe, NewReno, SACK (which are presently deployed, and to which we shall also refer to as the AIMD algorithm), Scalable, CuBIC, and HighSpeed TCP (which are proposed enhancements). The HighSpeed TCP algorithm proposes an increase and decrease algorithm which depends on the current transmission rate and the maximum available capacity. The authors provide two functions of the current transmission rate which are then used to compute the increments and decrements. As the name suggests, the CuBIC algorithm modifies the additive increase to a cubic increase function. The Scalable TCP algorithm proposes an even more aggressive multiplicative increase algorithm. An interesting feature of the multiplicative increase algorithm is that the time it takes to recover to the original window size after a packet loss is independent of the link capacity. This feature makes it a good candidate for implementation at a time when link capacities are on the rise. In addition to packet losses, the "delay-based" algorithms also use variations in the round trip delay to estimate the available bandwidth. A decrease in available bandwidth is interpreted as a sign of increasing congestion. The algorithms then reduce the rate accordingly. TCP Vegas (a "delay-based" additive increase additive decrease algorithm), FAST (a high speed version of the Vegas), and WestWood+ can be classified as "delay-based" algorithms.

Two of the desirable characteristics of a congestion control algorithm are efficiency and fairness. The fraction of the link capacity utilised by an algorithm is reflected in its efficiency. A higher efficiency translates into a lesser time to transfer the data, thereby resulting in higher user satisfaction. Usually, a link in a network is traversed by multiple flows originating from possibly distinct sources. Fairness reflects the ability of an algorithm to share the available capacity among the existing users in a "fair" way. Indeed, the exact definition of fairness among heterogeneous users[1] is arguable. For homogeneous users, we could safely designate an algorithm as fair if all the users receive equal share of the capacity. On a single link, the AIMD algorithm has been shown to be fair. The set of homogeneous users eventually (i.e., if the source stays long enough!) share the capacity equally. The heterogeneous users share the bandwidth in proportion to their round

[1]Heterogeneity can arise due to the use of different algorithms, or even due to the use of different parameters of the same algorithm.

trip times (RTT)[2]. Its efficiency, however, depends on the loss pattern (i.e., whether losses are synchronous or asynchronous, whether the asynchronous losses depend on the transmission rate or not), the available capacity, and the parameters of the algorithm. It may be agreed that any proposal to modify/replace the AIMD algorithm will need to have a higher efficiency and a similar, if not better, fairness property. These two characteristics can be classified as performance measures of an algorithm.

The performance measures of an algorithm can be studied using various methodologies. One way to study the performance of these algorithms is through simulations. The open source simulator *ns-2* provides the capability to evaluate the performance on various topologies and under different traffic conditions. Indeed, simulation studies can help to quantitatively understand the behaviour of any algorithm. A drawback of this method is the inability to exactly characterize the behaviour in terms of the parameters. The parameter space is often large, and to study the performance as a function of the parameters may require a large number of simulations. Another method to evaluate the performance is to setup a test-bed consisting of a network of machines which run the algorithm under review. Although similar to the method of simulations, this technique can give a more realistic behaviour of the algorithm. The methodology that we employ is that of stochastic modelling. The term stochastic modelling refers to models in which certain building blocks of the model are stochastic processes. A stochastic process is used to model the uncertain nature of an object. The drawback of such an approach is that models become analytically tractable only under certain simplifications. These simplifying assumptions need to be carefully studied and justified. On the positive side, these models can lead to some simple and elegant observations (for example, the "square root" formula for the average sending rate of the AIMD algorithm), and insights which can then be used to compare various algorithms. Stochastic models can also benefit from techniques of analysis which have been developed in other domains.

1.1 Motivation for the thesis

A major portion of this thesis is devoted to formulation and analysis of stochastic models for certain "loss-based" algorithms. The aim of such an analysis would be to obtain the related performance measures. The formulation and solution of a very general model incorporating all the nuances of a particular algorithm, which is subject to a general loss process, is quite a lofty ambition. Such a model will need to incorporate the interaction between multiple sessions and effect of traversing multiple links between the source and the destination. We set ourselves a more modest objective of modelling and analysing certain aspects of an algorithm. Even though simplifying assumption will be made, the analysis will hopefully shed some light on the performance aspects of these algorithms.

We shall mainly focus on the MIMD algorithm which has been suggested for Scalable TCP. As mentioned earlier, the MIMD algorithm has the property that the time to recover is independent of the capacity. Such a property is desirable during a time when the link capacities are constantly on the increase. The implementation of Scalable TCP also requires minimal changes to the present TCP. Below a certain threshold, Scalable TCP uses the additive increase algorithm. Beyond this threshold for the sending rate, Scalable TCP employs a multiplicative increase algorithm instead of the additive increase algorithm of the AIMD versions of TCP.

The first performance measure which we shall study is the average sending rate. Consider a single instance, or a session, of Scalable TCP which has large amounts of data to transfer. The instantaneous sending rate will depend on the increase algorithm, the decrease algorithm and the

[2]The interval of time from the moment a packet is sent to the moment its acknowledgement is received is called an RTT.

packet loss process seen by the source. Since the losses detected by the source occur at random, a stochastic model of the algorithm is desirable. The building blocks of the model shall be the increase algorithm, the decrease algorithm and the loss process. The loss process can be modelled in various ways. It could depend on the sending rate or be independent. It could be Markovian or identically distributed. One of the questions which we hope to answer is: how does the average sending rate depend on the loss process? The impact of the loss process on the sending rate will be explored in detail for the MIMD algorithm.

The other performance measure of interest is fairness. Fairness is usually quantified by a function of the sending rates, called the fairness index. The fairness index is used to define what a fair share is, and also quantifies how far a particular share is from the fair share. It has been shown that, for users of the MIMD algorithm, synchronous packet losses can lead to extreme unfairness. In other words, the fairness index can be arbitrarily far from the fair share value. We provide conditions on the loss process under which the fairness index for the MIMD algorithm can be brought closer to the fair share. One of the results obtained shows that the fairness index can be improved by introducing asynchronous packet losses at an intensity greater than a threshold intensity. These losses could be induced by a router using some active queue management (AQM) scheme[3]. If a proposed congestion control algorithm is implemented it is possible that it will be incrementally deployed. Thus, there may exist situations when MIMD and AIMD sessions will share a link. We will study how MIMD and AIMD flows traversing a common link share the available capacity when the packet losses are synchronous.

The "loss-based" algorithms react to packet losses by decreasing their sending rate. Packet losses usually occur when the network capacity is reached. For example, a router using the drop tail policy drops packets when its input buffer is full. It was felt that the drop tail policy was slow in reacting to congestion. Instead of waiting for the whole buffer to fill up, congestion signals[4] could be sent as soon as the buffer started to fill up. Queue management policies which send congestion signals in anticipation of congestion are called as AQM schemes. One such scheme, called random early detection (RED) [FJ93], probabilistically drops incoming packets. In this scheme, the router maintains an exponentially averaged queue length variable. An incoming packet is dropped with a probability which increases with increasing exponentially averaged queue length. We formulate a non-homogeneous QBD [Neu81] based model of the RED scheme, and analyse the model. The performance measures of interest for this study are the average queue length and the average drop probability.

In addition to stochastic models for congestion control algorithms, we also study two problems of interest in wireless networks. Wireless devices are usually powered by finite energy batteries which have to be recharged or replaced at the end of their lifetime. The energy that is delivered by a battery depends on how the battery is used. Continuous use of a battery leads to a much lower delivered energy compared to intermittent use. However, the device usually has packets to send, and intermittent use of the battery may add to the delay of packets. Our objective is to study the trade-off between energy and packet delay. The device has to decide whether to transmit packets or leave the battery idle so that it could recover some charge due to *relaxation phenomenon*. This decision problem can be formulated as a Markov Decision Process (MDP) [Put94]. The performance measure of interest is a cost function which needs to be minimized. This cost function comprises of a cost on the delay as well as cost on replacing/recharging the battery.

[3] An AQM scheme can be thought of as a congestion avoidance algorithm inside the network as opposed to TCP which is a congestion avoidance algorithm at the end-users [FJ93].

[4] A congestion signal could be either a packet drop or some explicit notification sent by means of setting bits in the acknowledgement packets.

Finally, we study a problem which arises in the UMTS cellular technology. Universal Mobile Telecommunications System (UMTS) is the third generation of wireless telecommunication technology which aims to provide voice and high speed data services. It envisages to replace the current GSM/GPRS architecture which provides services at a lower data rate. We study the average delay experienced by TCP sessions when they traverse a UMTS downlink. UMTS provides different channels to the users through which they can access services. The dedicated channel (DCH) has a higher data rate and a higher setup time as compared to the forward access channel (FACH). Scheduling a short session on the DCH can lead to larger delays for the session. However, the size of the session is not known in advance to the base station. Through simulations, we study the average delay performance of two policies which schedule the sessions based on the number of packets of the session which are in the transmit queue. The aim of these policies is to reduce the average delay experienced by each session. A larger queue length would indicate larger file sizes and thus it is more efficient to switch a session with a large queue length on to the DCH.

1.2 Organisation and contribution of the thesis

The research problems studied in this thesis can be divided in two parts. The first part comprises of Chapters 2-5 which are related to the performance analysis of congestion control algorithms and of RED. The second part comprises of Chapters 6 and 7 which are related to performance studies of certain problems in wireless networks.

In Chapter 2, we consider a discrete time model and study the behaviour of the window size of the MIMD algorithm when subject to various loss processes. The loss processes under consideration include the window-independent loss model, the window-dependent loss model, and the first-order Markovian window-independent loss model. We show that the logarithm of the window size process follows an additive recursive equation, which can be modelled as a Markov chain. The main results of this work are the expressions for the expected window size of an MIMD source when subject to window-independent losses. An approximate expression for the expected window size for window-dependent losses is also provided. The analytical results are then compared with simulations.

A continuous time model for a general increase algorithm and decrease algorithm is formulated in Chapter 3. It is shown that, by applying a certain transformation, a general algorithm can be studied as a general increase and additive decrease algorithm with a loss process appropriately transformed as well. The transformation to the additive decrease algorithm will enable us to relate the window size with the workload in a queueing system. By means of this analogy, we provide conditions under which two congestion control algorithms will have the same behaviour. For example, it is shown that the AIMD algorithm with a window-independent loss process has similar behaviour as the MIMD algorithm with window-dependent loss process.

The fairness properties of the MIMD algorithm are studied in Chapter 4. We study the fairness properties for both *homogeneous* and *heterogeneous* users. First, we study the *intra-algorithm*[5] fairness property of the MIMD algorithm. The ratio of the instantaneous rates of two users is modelled as a Markov chain which is either transient, null recurrent or positive recurrent depending upon the loss process and its intensity. For homogeneous users, it is shown that the fairness index of the MIMD algorithm can be improved by introducing asynchronous losses at a non-zero intensity. For heterogeneous[6] users, it is shown that the intensity of asynchronous losses should be greater than a threshold intensity. Next, we study the *inter-algorithm* fairness properties of

[5]By *intra-algorithm* we mean the scenario when all users employ the same algorithm.

[6]The heterogeneity is due to different round trip times.

the MIMD algorithm. Specifically, we study how MIMD and AIMD users share the link capacity when they are subject to synchronous losses. We show that the bandwidth share obtained by the AIMD users is independent of the link capacity (assuming the capacity is large enough), and that the MIMD users utilise the rest of the capacity.

In Chapter 5, we model the dynamics of a RED queue driven by Poisson traffic. For a router with a large link capacity, a small averaging parameter has been proposed [FGS01]. Since the averaging parameter is small, the exponentially average queue and the instantaneous queue evolve on different time scales. The time-scale separation allows us to use a singular perturbation based approach to analyse the behaviour of the average and instantaneous queue lengths. We use a discretization method by means of which we model the joint average and the instantaneous queue length as non-homogeneous quasi birth-and-death (QBD) process. In the limit when the averaging parameter goes to zero, we use a singular perturbation approach to obtain the stationary distribution for the joint queue length process, and the average drop probability of the RED algorithm.

The energy-delay trade-off for a wireless device is studied in Chapter 6. We formulate the problem as a discrete-time Markov decision process. We study the finite horizon discounted cost, the infinite horizon discounted cost and the average cost problems. Under certain conditions on the cost function, it is shown that the optimal transmission policy is of threshold type, i.e., the device transmits packets only if the queue length exceed the threshold. The optimal policy is also shown to be bang-bang type, i.e., the device either transmits the maximum possible number of packets or transmits nothing.

Finally, the average delay of TCP sessions for two threshold based channel selection policies in UMTS is evaluated in Chapter 7. We use simulations as a means of comparing the performance of the two algorithms.

The conclusions and the possible further research directions of this work are presented in Chapter 8.

Chapter 2

Performance analysis of the MIMD algorithm in discrete time

In this chapter, we study the behaviour of the window size of the MIMD congestion control algorithm when it is subject to different loss processes. We start by considering the loss process to be independent of the current window size. The window size observed at certain time instants (at RTTs) is modelled as a multiplicative recursive equation. The logarithm of the window size is shown to have the same behaviour as the workload process in a standard G/G/1 queue. The Laplace-Stieltjes transform of the equivalent queue then directly provides the throughput of the congestion control algorithm and the higher moments of the window size. Using the analysis for the window-independent loss process, an approximation is proposed for the window-dependent loss process. Finally, the performance measures are studied for a window-independent first-order Markovian loss process. We validate our findings using Scalable TCP in *ns-2* simulator.

The results in this chapter have appeared in [AAB+05].

2.1 Window based congestion control

We begin with a brief description of the functioning of a window based congestion control algorithm.

A window-based congestion control algorithm attempts to reliably transfer data and to avoid congestion within the network. Let us assume that the data to be transferred is divided into equal size blocks called packets.

Let us consider a source which wants to send an infinite number of data packets to a destination, as is shown in Figure 2.1. The source and the destination are connected through a network[1] whose transmission capacity[2] is unknown to both the source and the destination. A window-based congestion control algorithm attempts to estimate the rate at which it can send the packets through the network. Although the link capacities are fixed, there are other users who make use of the network. Since a congestion control algorithm has to be fair and efficient, it will need to adapt its

[1] A network can be considered to be a graph with a set of vertices and a set of edges. The vertices are routers which forward incoming packets to on one of the outgoing edges. The edges are finite transmission rate links which connect two routers in the network.

[2] Transmission capacity is the amount of data that can be transferred in a unit of time.

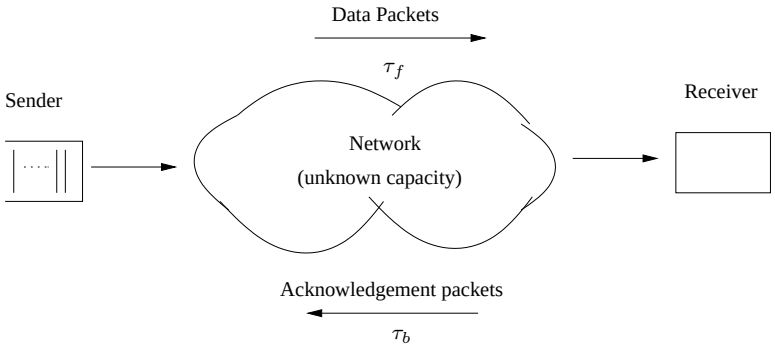


Figure 2.1: Reliable data transfer through a network.

sending rate as a function of number of users present in the network. If other users leave the network, the algorithm should increase its rate so as to efficiently use the link capacity. And, if users enter the network, then the algorithm should decrease the sending rate so as to give a fair share to the incoming users.

In order to adapt the sending rate, the algorithm needs information on the utilisation of the network. This information is sent from the destination to the source by means of acknowledgement packets. As soon as the destination receives a data packet, it sends an acknowledgement (ACK) packet back to the source. The main purpose of ACK packet is to inform the source of faithful (or, at least what it thinks is faithful) reception of a particular data packet at the destination. Additionally, the ACK packets can also convey information on the level of congestion in the network[3]. The interval of time from the moment a packet is transmitted by the source to the time the corresponding acknowledgement is received by the source is called a round trip time (RTT)[4]. The RTT comprises of the time required for propagation of the packets on the links and the time spent in the buffers of the routers.

In order to ensure reliable packet delivery, the source maintains a list of data packets which it has transmitted but for which it has not yet received an acknowledgement. This list, called a window, contains the data packets which the source presumes to be in the network. At a given time t, the window size determines the number of ACK packets which the source can expect to receive in the interval $[t, t + RTT]$. Thus, the number of packets in this list, or the window size, is proportional to the sending rate[5]. The source adapts the sending rate by either increasing or decreasing the window size.

From the algorithm's perspective (shown in Figure 2.2), the network can be simplified to a

[3]This is done by setting bits in the ACK packet. This type of feedback is called the explicit congestion notification (ECN) as opposed to the implicit congestion notification (ICN) in which the source infers congestion from some information (for example, a packet loss). Indeed, ECN is more representative of the network conditions than ICN. For, ICN may be based on information which may not be related to congestion at all. For example, ICN infers every packet loss as a sign of congestion. A packet loss could occur not only due to congestion but also due to errors on the wireless channel or on the optical fibre. This causes the algorithm to react even though there is no congestion.

[4]In practice, the RTT is itself a function of time, but for the purpose of presentation we shall make the simplification that the RTT is a constant.

[5]This observation is valid only when the RTT is a constant. If a large number of packets of the source are queued up in a router buffer then the RTT itself increases with the window size. Although the source sends packets at a higher rate by increasing the window size, the rate of receiving the ACK packets (i.e., the effective service rate) cannot be larger than the rate of the smallest capacity link on the path. The effective sending rate, or the throughput, of the source is thus bounded.

black box which has a transmission capacity $c(t)$ (the targeted fair share of the algorithm), and which gives feedback signals at rate $h(t)$ (the congestion information delayed by an RTT). The algorithm adapts its sending rate $r(t)$ (i.e., its window size) so as to be as close as possible to $c(t)$. A model of a window-based algorithm would consist of the increase algorithm (to be used when no congestion is perceived in the network), the decrease algorithm (to be used in reaction to increasing congestion), and the congestion signals, $h(t)$ (which indicate the level of congestion).

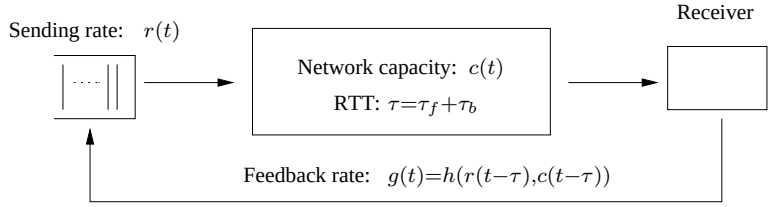


Figure 2.2: The network as perceived by a source.

2.1.1 The increase algorithm

In the absence of congestion signals, the source presumes the network capacity is not fully utilised. Hence, in order to be more efficient, the source employs the increase algorithm.

For each positive acknowlegdement, i.e., in the absence of congestion, the source increases its window size in a deterministic way. The dynamics of the window size[6] can be described by the recursive equation

$$w_{n+1} = \min(f(w_n, n), W_{max}), \tag{2.1}$$

where $f(x, n) \geq x$, W_{max} is the maximum window size which we shall define in Section 2.1.2, and the time index n denotes the reception of the nth acknowlegdement (positive or negative). For example, in the AIMD algorithm, for each positive acknowledgement the window size is updated as

$$w_{n+1} = \min(w_n + c/w_n, W_{max}). \tag{2.2}$$

Since w_n is also the number of data packets transmitted during the previous RTT, the source can expect to receive w_n ACK packets during the present RTT. Hence, when there is no congestion, the source increases its window size by c at time intervals of RTT. This translates into an additive increase of the window size sampled at intervals of RTT. For the MIMD algorithm, the function f is linear in w, which translates into a multiplicative increase of the window size sampled at intervals of RTT.

For the purpose of theoretical modelling, the window-based algorithms can be modelled in continuous time. The data is modelled as a fluid which can be transmitted in infinitesimal quantities. Unlike in the physical system where the source adapts the window size at specific points in time (i.e., on the reception of ACK packets), in this model the window size is adapted in continuous time (i.e., the feedback from the destination is also assumed to be a fluid which is flowing at all instants in time.) The dynamics of the window size during the increase phase can described by the differential equation

$$\dot{r}(t) \quad = \quad \begin{cases} f(r(t), t), & r(t) \leq r_{max}, \\ 0, & r(t) > r_{max}, \end{cases}$$

[6]The window size could possibly be non-integer. The number of outstanding data packets is, however, always an integer and it is determined by the integer part of the window size.

where f is a positive function. For example, the additive increase algorithm can be modelled by taking f to be a constant equal to c/RTT.

By increasing its sending rate the algorithm probes the network for its fair share.

2.1.2 Congestion notification

Congestion occurs at a router when the net input rate for an outgoing link exceeds the maximum transmission rate at that outgoing link. Since any link has a finite maximum transmission rate, the increase algorithm ensures that congestion will eventually occur[7]. The excess input rate causes the buffer at the router to fill up. The maximum window size that the source can sustain without causing the queues in the network to build up is called the *bandwidth-delay product* (BDP) of the user. The BDP corresponds to the maximum number of outstanding packets the source can have in the network. As the name indicates, the BDP is computed by using the relation[8]

$$\text{BDP(packets)} = \frac{(\text{max. transmission capacity(bits/sec)}) \times (\text{min. round trip delay(sec)})}{\text{packet size (bits)}}. \quad (2.3)$$

At the onset of congestion, the router can explicitly notify the level of congestion by setting bits in the data packets. These bits are then sent back to the source by the destination by means of the ACK packets. It is possible that the ECN mechanism may not be implemented in some of the routers. These routers either allow the buffer to overflow (drop tail queue management) or drop packets from the buffer using some active queue management (AQM) scheme.

Since a packet will be dropped once the router buffer is full, the source will perceive a packet loss (delayed by an RTT) when the window size exceeds the sum of the BDP and the input buffer size at the router[9]. Additionally, the destination may also have resource constraints such as a finite receive buffer. It may therefore request the source to limit the maximum window size to a value called the *receiver's advertised window*, W_{recv}. Let $W_{cong} = BDP + buffer\ size$ denote the maximum window size that the source can sustain without causing any packet drops. The maximum window size, W_{max}, of the source is limited by

$$W_{max} = \min(W_{cong}, W_{recv}). \quad (2.4)$$

However, packet drops are certain to happen only when $W > W_{cong}$.

As the router buffer starts to fill up, the RTT measured by a source using this router also starts to increase. This increase in RTT is interpreted as a sign of congestion by "delay-based" congestion control algorithms. The "loss-based" congestion control algorithms rely only on packet losses [10] to signal congestion. The "loss-based" algorithms are said to rely on binary feedback signals. The function $h(t)$ either indicates congestion or indicates no congestion in the network.

[7] This may not be true if the source has a finite number of packets to send.

[8] In this relation, we assume all the packets are of equal size, and compute the BDP in units of packets. In general, the BDP could be computed in bits by using only the numerator of the given relation.

[9] Beyond this window size, a packet loss will inevitably occur. However, if there are other users sharing the network, or if there are random losses, then packet losses may occur earlier.

[10] The data packets sent by the source contains a unique identifier called the sequence number. The destination expects to receive packets in order of the sequence number. For each data packet received, the destination sends back an ACK packet containing the sequence number of the last data packet received in sequence. The source infers the loss of a packet when it receives more than one ACK packet with the same sequence number.

There is no information on the extent of congestion within the network[11]. In this thesis, we only study congestion control algorithms which are "loss-based".

The feedback function $h(t)$, which takes only binary values, can also be modelled in discrete or continuous time. In continuous time, $h(t)$ can be modelled as a point process [BB03] (the points indicating the arrival instants of loss notifications). The inter-arrival time of these points could be independent of the current window size or could depend on the window size. For example, the process $h(t)$ could be a Poisson point process with intensity λ.

The "loss-based" algorithms react to losses irrespectively of the origin of the packet loss which could be either due to congestion or due to some random event. We shall use the term "congestion loss" to denote a loss which occurs when the input rate to the network exceeds its capacity (i.e., $W > W_{cong}$). The term "random loss" will denote a packet loss due to any other event. Henceforth the term "loss" will denote a generic loss event which could be due to either congestion or a random event.

2.1.3 The decrease algorithm

The arrival of loss (not only congestion) notifications at the source triggers the decrease algorithm. In discrete time systems, on the reception of a negative acknowledgement, the window is updated as

$$w_{n+1} = \max(g(w_n), W_{min}), \tag{2.5}$$

where $g(x) < x$, and W_{min} is the minimum window size, usually set to unity. For the multiplicative decrease algorithm, the source decreases the window size by a factor $1 - \beta$, where $\beta < 1$, of the current window size, i.e.,

$$w_{n+1} = \max(\beta w_n, W_{min}), \tag{2.6}$$

For the additive decrease algorithm, like in TCP Vegas, the window is reduced by a constant number.

In continuous time systems, the decrease algorithm can be modelled as an instantaneous jump in the window size,

$$r(t+) = \max(g(r(t)), r_{min}), \tag{2.7}$$

where $g(x) < x$.

2.2 The standard TCP algorithm

In the preceding section we described the functioning of a general window based congestion control algorithm. The transmission control protocol is a window-based congestion control algorithm which is widely deployed in the end hosts of the Internet. The TCP algorithm, which we shall call as the standard TCP[12], has its origins in the algorithm proposed by Van Jacobson [Jac88]. We now give a simplified description of the actual implementation (described, among others, in [APS99]), and motivate why this algorithm was found to be inefficient in networks which provide large transmission rates.

[11] The ECN scheme proposed in [Flo94], which sets only one bit (say, 0 indicating no congestion, and 1 indicating congestion), is also an example of binary feedback.

[12] The TCP algorithm has evolved over the years. The algorithm which we will describe is closer to the NewReno version.

The TCP algorithm operates in four phases: *slow start, congestion avoidance, fast retransmit*, and *fast recovery*. The source starts by sending one data packet[13] (i.e., window size of one) to the destination. For each positive ACK packet received at the source, the window size is increased by one. Thus, the sending rate increases multiplicatively in this phase. The fast increase in sending rate ensures that the source is quickly able to estimate its fair share. This is the *slow start* phase, and the source remains in this phase until it either experiences a loss or the window size reaches a threshold (called the *slow start threshold*) or the window size reaches the maximum window size requested by the destination, W_{recv}.

The source infers a loss when it receives four ACK packets acknowledging the same data packet[14] or a timer, known as the retransmission timer, expires. The retransmission timer is reset to a given value[15] whenever either a new data packet is sent and the timer is off or a new ACK packet is received. Since the source sends new packets only when an ACK packet arrives, the timer prevents the source from remaining idle for too long in case there is an interruption in the ACK stream.

Once a loss is detected using duplicate ACKs, the algorithm retransmits the lost packet(s)[16] and decreases the window size by half. This is the *fast retransmit phase* which is immediately followed by the *fast recovery* phase which lasts until a positive ACK is received for the lost packet. In the *fast recovery* phase, the source transmits a new data packet for each ACK packet which is received. The arrival of ACK packets indicates that the data packets are indeed getting across to the destination, thereby freeing up the network resources.

Once the ACK packet for the retransmitted data packet is received, the algorithm enters the *congestion avoidance* phase. On reception of each new ACK packet, the algorithm increases the window size as

$$w_{n+1} = \min(w_n + 1/w_n, W_{max}). \tag{2.8}$$

However, the window size cannot increase beyond the maximum window size requested by the receiver. On the reception of duplicate ACKs, the algorithm exits the *congestion avoidance* phase and enters the *fast retransmit* phase. If there are no timeouts then this cycle of alternating *congestion avoidance* and *fast retransmit/recovery* phases continues until all the data has been correctly received by the destination. If the retransmit timer expires, the source sets the *slow start threshold* to half the current window size and then resets the window size to the minimum window size, W_{min}(which is usually equal to unity), and enters the *slow start* phase.

Let W_n be the window size of the source at end of the nth RTT. Assuming there are no timeouts, the update equations for W_n are given by[17]

$$\begin{array}{rcll} W_{n+1} & = & \min(W_n + 1, W_{max}) & \text{if there are no losses;} \\ W_{n+1} & = & \max(0.5 \cdot W_n, W_{min}) & \text{if there are one or more losses.} \end{array} \tag{2.9}$$

Therefore, the standard TCP uses the AIMD algorithm. Chiu and Jain [CJ89] studied the fairness

[13] Sometimes the source can decide to start at a larger initial window size.

[14] The ACK packets (except for the first one) acknowledging the same data packet are called duplicate ACKs.

[15] This value is computed as a function of the estimated mean and deviation of the RTT. The estimates for mean and deviation are computed by using a low-pass filter on the ACK stream.

[16] The selective acknowledgement (SACK) version of TCP [MMFR96] permits the destination to explicitly inform the source of the data packets which are lost. In case of multiple packet losses, the source reduces the window size only once and retransmits only the lost packets.

[17] We assume that SACK is used.

and efficiency properties of congestion control algorithms of the form

$$\begin{array}{rcll} W_{n+1} & = & a_I + b_I W_n & \text{if there are no losses,} \\ W_{n+1} & = & a_D + b_D W_n & \text{if there are one or more losses,} \end{array} \tag{2.10}$$

when all the sources sharing a link were subject to a loss at the same time instant. They concluded that the AIMD algorithm ($a_I > 0$, $b_I = 1$, $a_D = 0$, $b_D < 1$) required the minimum time to converge to fairness. Their conclusion on the AIMD algorithm was also one of the reasons why the AIMD algorithm was recommended for standard TCP by Van Jacobson in [Jac88].

2.3 The Scalable TCP algorithm

The standard TCP algorithm was implemented during a time when the typical average sending rates obtained by end hosts was of the order of kilobytes per second (kBps). The link capacities in Internet have increased rapidly during the past few years, providing the end hosts with average sending rates of the order of megabytes per second (MBps). Although the fairness properties of the AIMD algorithm are very desirable, the additive increase algorithm does not recover rapidly enough after a loss which is not due to congestion [FRS01]. It was generally felt in the Internet research community that the standard TCP algorithm needed to be modified so that it could perform satisfactorily in large capacity networks as well. Since then, several new algorithms have been proposed in [Flo03], [JWL04], [XHR04], [Kel03b] and [KHR02] among others.

In [Kel03b], T. Kelly proposed an algorithm, called Scalable TCP, which modified the additive increase during the *congestion avoidance* phase to a multiplicative increase. On the reception of a new ACK packet, the Scalable TCP algorithm increases the window size as

$$w_{n+1} = \min(w_n + 0.01, W_{max}). \tag{2.11}$$

Since the source can expect to receive w_n ACK packets in the $(n+1)$th RTT, the embedded window size will increase multiplicatively in the absence of congestion. The decrease algorithm and the other features (like, for example, the retransmission timer) of standard TCP are preserved in Scalable TCP. The author also introduces the notion of a legacy threshold for the window size. If the current window size is below the legacy threshold, Scalable TCP[18] behaves like the standard TCP. Since the multiplicative increase algorithm is faster than the additive increase algorithm, the legacy threshold ensures that in small capacity networks the standard TCP users are not penalized.

Let W_n[19] be the window size of the source at the end of the nth RTT. In the *congestion avoidance* phase, the MIMD algorithm[20] updates W_n as

$$\begin{array}{rcll} W_{n+1} & = & \min(\alpha \cdot W_n, W_{max}) & \text{if there are no losses,} \\ W_{n+1} & = & \max(\beta \cdot W_n, W_{min}) & \text{if there are one or more losses.} \end{array} \tag{2.12}$$

where $\alpha > 1$ and $\beta < 1$ are constants. For the Scalable TCP algorithm, the values $\alpha = 1.01$ and $\beta = 0.875$ have been proposed by T. Kelly [Kel03b].

The Scalable TCP algorithm has the following desirable property. Let us assume that a source with window size w experiences a packet loss due to reasons other than congestion. Since

[18] We shall use the term MIMD algorithm interchangeably with Scalable TCP, and the term AIMD algorithm interchangeably with the standard TCP.

[19] We shall use this notation henceforth.

[20] We assume that Scalable TCP with SACK is used.

the TCP (either standard or Scalable) algorithm cannot discern the cause of the loss, the source will reduce the current window size to βw. The time T_r (in number of RTTs) required for the source to increase the window size from βw to w will be

$$\begin{aligned} T_r &= (1-\beta)w \quad \text{for standard TCP,} \\ T_r &= -\log_\alpha \beta \quad \text{for Scalable TCP.} \end{aligned}$$

During the interval T_r, the source sends data at a rate which is less than its fair share. From the above equations we can conclude that the time to recover, T_r, for the AIMD algorithm is proportional to the window size (and, by consequence, to the network capacity) whereas it is independent of the window size for the MIMD algorithm. As the network capacity increases, the AIMD algorithm will become inefficient in utilising the available capacity. On the other hand, for the MIMD algorithm, the time to recover will be unaffected by an increase in the network capacity, thus making it "Scalable".

2.4 A discrete time stochastic model for the MIMD algorithm

The main emphasis of this chapter is to obtain certain performance measures, such as the average sending rate, of the MIMD algorithm when it operates in different environments. Towards this end, we first present a discrete time stochastic model for the window size evolution of the MIMD algorithm. We shall then adapt this model for different environments, and obtain the performance measures in each of them.

Let us consider an MIMD source which has an infinite amount of data packets to send[21]. Let W_n denote the window size at the end of the nth RTT. We assume that the round trip time is a constant. This is an approximation to the scenario where the queueing delay is much less than the propagation delay.

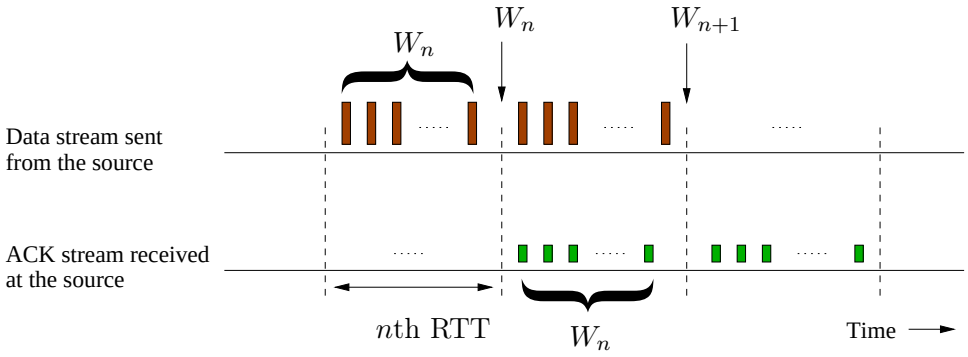


Figure 2.3: The Window evolution in discrete time.

During the $(n+1)$th RTT, the source will receive the acknowledgements for the data packets sent during the nth RTT. Let $p_n(W_n)$ denote the probability that one or more data packets were lost during the nth RTT. If we assume there are no timeouts, the window size will evolve according to the recursive equation

$$\begin{aligned} W_{n+1} &= \min(\alpha \cdot W_n, W_{max}) \quad \text{with probability (w.p.)} \quad 1-p_n(W_n), \\ W_{n+1} &= \max(\beta \cdot W_n, W_{min}) \quad \text{with probability} \quad p_n(W_n), \end{aligned} \tag{2.13}$$

[21] Such a source-destination pair is also called a persistent connection.

where $\alpha > 1$ and $\beta < 1$ are known parameters. In our model we have assumed that there is either an increase or a decrease during an RTT. In practice, even if a packet loss is observed in an RTT, the window size increases for all the positive ACKs which are received. For example, in the $(n+1)$th RTT, if the ith ACK packet signals a lost data packet then W_{n+1} would be equal to $(\beta+(\alpha-1))W_n-(i-1)(\alpha-1)(1-\beta)-(\alpha-1)$. Since i takes values from unity to W_n, the window size W_{n+1} lies in the interval $(\alpha\beta W_n-(\alpha-1),(\beta+(\alpha-1))W_n-(\alpha-1)]$. Thus, the approximation $W_{n+1}=\beta W_n$ is a lower bound for the true value of W_{n+1}. The recommended values, $\alpha=1.01$ and $\beta=0.875$, allow us to closely approximate W_{n+1} by βW_n.

2.4.1 The loss process

The loss process seen by the source is highly influenced by the environment in which the source operates. The number of links on the route from the source to the destination, the number of sources sharing each of these links, and the type of the links (whether the link is an optical fibre, or a wireless channel, or Ethernet cable) are some of the factors which determine the loss process. A loss process is usually determined by collecting the statistics of the losses observed by a source in different environments [AAB00a]. We shall use three different models for the loss process. These models have been used in the past to obtain the average sending rate of the standard TCP algorithm. These loss models are approximations to different environments in which a source may operate.

1. *Window-independent loss process*: In this model, the probability of loss in an RTT, $\{p_n(W_n), n \geq 0\}$, is an independent and identically distributed stochastic process and is independent of the current window size of the source. The recursive equation for W_n is given by

$$W_{n+1} = \begin{cases} \min(\alpha \cdot W_n, W_{max}) & \text{w.p.} \quad 1-p; \\ \max(\beta \cdot W_n, W_{min}) & \text{w.p.} \quad p, \end{cases} \tag{2.14}$$

   where p is the probability of experiencing one or more losses in an RTT. We shall use the term *loss rate*[22] to refer to the probability of experiencing a loss in an RTT.

   The independence has been observed in connections over wide area networks [AAB00a], and has also been studied in [BH02]. In [MGT99], the loss process is modelled as a Poisson point process with a fixed rate. The independence is observed in [MGT99] when a large number of a flows share a link and a flow traverses multiple links.

2. *Window-dependent loss process*: In some environments, each packet has a certain probability of either being dropped or lost. For example, packets can be lost due to imperfections in an optical fibre cable. Certain AQM schemes drop packets randomly from the buffer [PPP00]. Thus, a source with a larger window size has a larger probability of having packets in the buffer, and is thereby more prone to experiencing a loss.

   We assume that each packet is lost with a probability q independently of the other packets. This model was first used in [OKM96] to derive the well-known "square-root" formula for the throughput of the AIMD algorithm. In this model, the process $\{p_n(W_n), n \geq 0\}$, depends on

[22] Since p is the probability of loss in one RTT and the discrete time is in units of RTT, p can be considered as the probability of loss per unit time. Hence, we refer to p as the loss rate. The term loss probability will be used to refer to packet loss probability.

W_n but is independent from one RTT to another. The recursive equation for W_n is given by

$$W_{n+1} = \begin{cases} \min(\alpha \cdot W_n, W_{max}) & \text{w.p.} \quad (1-q)^{W_n}; \\ \max(\beta \cdot W_n, W_{min}) & \text{w.p.} \quad 1-(1-q)^{W_n}. \end{cases} \tag{2.15}$$

3. *Window-independent first-order Markovian loss process*: The first-order Markovian loss process allows us to introduce correlation among the losses experienced in successive RTTs. The loss rate will be modelled as a first-order Markov chain which transits at RTTs (cf. [AAB00b] in which the transitions occur at loss instants.). The probability of loss in the nth RTT will be considered to be independent of W_n. The recursive equation for W_n is given by

$$W_{n+1} = \begin{cases} \min(\alpha \cdot W_n, W_{max}) & \text{w.p.} \quad 1-p_n; \\ \max(\beta \cdot W_n, W_{min}) & \text{w.p.} \quad p_n, \end{cases} \tag{2.16}$$

where $\{p_n, n \geq 0\}$ is a finite dimensional first-order Markov chain.

We briefly mention, but do not study, another model for the loss process which is particularly suited for wireless links. In this model, known as the Gilbert-Elliot model [WM95], the packet loss probability, q_n, is a first-order Markov chain. In our description of loss processes, this model would be described as *window-dependent first-order Markovian loss* process. The performance study of the standard TCP algorithm using a Gilbert-Elliot model has been carried out in [KH98], [ZCR00] and [AT03].

2.4.2 The time average service rate

A performance measure of interest is the time average sending rate. The time average service rate, or the throughput, is defined as

$$\gamma = \lim_{t \to \infty} \frac{\text{number of packets sent during } [0,t]}{t} \tag{2.17}$$

Let $\tau(W_n)$ denote the length of the nth RTT, i.e., the nth RTT is a function of the window size, W_n. From the Markov renewal-reward theorem [Wol89], the throughput can be obtained as

$$\gamma = \frac{\mathrm{E}[W]}{\mathrm{E}[\tau(W)]}, \tag{2.18}$$

where $\mathrm{E}[f(X)]$ denotes expectation of a function f of a random variable X. In the discrete time model, we embed the window size process at intervals of RTT, which is assumed to be constant. Thus, we can obtain the observed throughput by using the relation $\gamma = \mathrm{E}[W]/\mathrm{RTT}$.

2.4.3 A few words on modelling assumptions

There are various aspects of a practical congestion control algorithm which we have not incorporated in our model. In the parlance of the research community, we model an "idealised" MIMD congestion control algorithm (see, e.g., [MO99], which considers the "idealised" behaviour of the AIMD algorithm).

1. *No timeouts:* The first approximation that we make is that there are no timeouts observed by the source. The effect of a timeout is to reduce the current window to W_{min}. Timeouts usually occur when a source experiences multiple packet losses. We assume the Scalable TCP algorithm is used along with SACK which allows the source to recover from multiple losses. Although a timeout can still occur, the probability is lesser than the other versions (like Tahoe or Reno) of TCP. Our model, therefore, gives an upper bound to the average sending rate which becomes tighter as the probability of timeouts goes to zero.

2. *At most one multiplicative decrease per RTT:* We also assume the window is reduced only once during an RTT, which is also a feature of the SACK version.

3. *Infinite file size:* This assumption is frequently made in the analysis the behaviour of congestion control algorithms. Modelling short file sizes is rather difficult. Also, the application layer protocol *http 1.1* [FGF+99] can transfer several files in a batch over the same TCP connection, thus getting the performance of a single large file. We assume that files are large enough to see a regime close to stationarity.

4. *Constant round trip times:* The constant RTT assumption is quite important in relating the average window size and the average service rate (or, the throughput) that a connection obtains. The RTT is the sum of the propagation delay in the links and the queueing delay in the routers. With the constant RTT assumption, the throughput can be directly obtained as $\mathrm{E}[W]/RTT$. In networks with small buffering delay (in units of time required to empty the buffer) compared to the propagation delay, this assumption is a good approximation. As the buffering delay increases, more and more packets start getting queued up in the buffer causing the round trip times to increase. The RTTs will now be closely linked to the window size, Since the service rate of a link is finite, the throughput obtained by the source will be upper bounded by this quantity. The effect of a non constant RTT is to modify the increase algorithm, see, for example, [LM97] and [AABQ01b].

5. *No legacy threshold:* Our main aim is study the behaviour of the MIMD algorithm. For low error probabilities and large capacity links, this assumption would become more realistic.

2.5 Outline and contribution

In Section 2.6, we shall give a summary of the various analytical techniques which have been employed to obtain the throughput of the standard TCP algorithm. In Section 2.7 we shall analyse the model with the *window-independent loss* process. Our approach, formally described in Section 2.7.1, is based on showing that an invertible transformation applied to the window size process results in a process that has the same evolution as the total workload process in a standard G/G/1 queue. The Laplace-Stieltjes transform of the equivalent queueing process thus obtained provides the throughput of the connection as well as the higher moments of the window size of the given MIMD algorithm. In Section 2.8, we shall study the model with the *window-dependent loss* process. We shall propose an approximation to this model, and obtain the throughput of the connection using the analysis of the *window-independent loss process.* In Section 2.9 we shall study the *window-independent first-order Markovian loss* process and obtain the performance measures of the window size. We shall compare the analytical results with simulation results in Section 2.10.

2.6 Related literature

Since it was proposed in 1988 [Jac88], the behaviour of standard TCP has been widely studied. We cite only a few articles which are related to this work. In [BHCKS04], a good overview of the analytical methods for studying the window size behaviour of standard TCP, i.e., for the AIMD algorithm, is provided.

One of the first stochastic models of the AIMD algorithm was proposed in [OKM96]. The authors studied the window size observed at arrival of ACK instants. For a packet loss probability of q, it was shown that the throughput was proportional to $1/\sqrt{q}$. In [LM97] the authors studied the behaviour of two different flavours of standard TCP when the delay due to link buffer is not negligible. They considered a deterministic loss model and obtained an approximation when the losses were random. Their model was extended in [Kum98] to include the effect of timeouts. The window size was modelled as a Markov chain embedded at loss instants or arrival of timeouts. The throughput was obtained as the reward rate of the Markov renewal-reward process. In [KH98], this model is extended to include the Gilbert-Elliot loss model for the packet error probability. In [PFTK98], the authors considered a model in which they accounted for the number of packets sent between the loss instant and the reception of the duplicate ACKs. For their model, which also included timeouts, they derive an expression for the throughput of a TCP session when packets are lost with a probability q and validate with experimental results. In [AAB00a] and [AAB00b] the authors study the performance of TCP when the loss process is a general stationary point process which may depend on the window size. In particular, they obtain the throughput expressions when the inter-loss time is an i.i.d. renewal process and when loss process is a Markovian Arrival Process(MAP). In [AABQ01a], the authors study a continuous time model wherein the window size grows linearly between loss arrivals. The losses are assumed to arrive in batches according to a Poisson process which is independent of the window size. For both bounded and unbounded window sizes they obtain the distribution of the window size and its moments. In [BHCKS04], the authors consider a similar formulation but with a loss intensity which is a linear function of the window size, and they also account for timeouts.

The above cited articles mainly consider the behaviour of a single source which has a large amount of data to send. The interaction among various flows has been modelled in, for example, [BH02] and [MGG$^+$04], which also has references related to multiple flow models. In [MSZ02] and [CSA01], the performance of flows that have finite amount of data to send is analysed.

For deterministic losses, the authors considered a general increase and decrease algorithm in [BB01]. In an RTT without a loss, the window increases by w^{-k} whereas in an RTT with losses, the window reduces by w^l. For $k = 0$ and $l = 1$ the algorithm becomes AIMD, and for $k = -1$ and $l = 1$ the algorithm becomes MIMD. They obtain the result that the throughput for such algorithms is proportional to $q^{-(k+l+1)}$. For example, for the MIMD algorithm the throughput becomes proportional to q^{-1}. We shall also obtain the same proportionality by first analysing a random drop model for independent losses and then using an approximation of small loss probability.

2.7 Window-independent loss process

Let us consider a source which wants to transfer a large amount of data using the MIMD algorithm. The source starts by sending $W_0 = W_{min}$ amount of data across the network and waits for the corresponding ACK packets. Let t_0 denote the time instant when the last of the W_0 packets is transmitted. At t_0, the number of outstanding packets, i.e., the window size is W_0. Let t_1 denote

the time instant just after the source receives the W_0th ACK packet (i.e., at t_1 the algorithm has either increased or decreased the window depending on the W_0th ACK packet.). At this time instant the window size would be either $W_1 = \alpha W_0$ or $W_1 = \beta W_0$. Similarly, let t_2 denote the time instant when the $(W_0 + W_1)$th ACK packet is received, and let W_2 be the window size at this time instant. In this way we can construct the sequence $\{(W_n, t_n), n \geq 0\}$. We shall call the interval $(t_{n-1}, t_n]$, for $n \geq 1$, as the nth RTT. Let W_n denote the window size at the end of the nth RTT. During the nth RTT the source sends W_n packets, and can expect to get the corresponding ACK packets in the $(n+1)$th RTT.

Let the sequence $\{A_n, n \geq 1\}$ be defined as

$$A_n = \begin{cases} \alpha & \text{w.p.} \quad (1-p); \\ \beta & \text{w.p.} \quad p, \end{cases} \tag{2.19}$$

where p is the probability of one or more losses occurring in any RTT. As described in Section 2.4, we can model the process $\{W_n\}$ by the recursive equation

$$W_{n+1} = \min(\max(A_{n+1} W_n, W_{min}), W_{max}). \tag{2.20}$$

We now adapt the above equation to three different scenarios so that the simplified equation will help us to better understand the process $\{W_n\}$ in each of the three scenarios.

(i) No upper bound on window size: In the first scenario, we assume there is no upper bound, i.e., $W_{max} = \infty$. Equation (2.20) simplifies to

$$W_{n+1} = \max(A_{n+1} W_n, W_{min}). \tag{2.21}$$

In practice, as described in Section 2.1.2, W_n is limited by either the receiver's advertised window or the network capacity. However, if the loss rate is sufficiently large such that the upper bound is rarely reached, then this equation can be used to approximate the true window behaviour. We shall analyse this scenario in Section 2.7.2.

(ii) No lower bound on window size: On the other hand, if the losses are infrequent, then the probability of reaching the lower bound will be negligible. In such a scenario, we can assume the lower bound to be zero and obtain the recursive equation

$$W_{n+1} = \min(A_{n+1} W_n, W_{max}). \tag{2.22}$$

The upper bound in this scenario corresponds to limitation imposed by the receiver's advertised window (i.e., $W_{recv} \leq W_{cong}$). The receiver's advertised window is assumed to be below the network capacity. Thus, there are no losses due to congestion. This scenario will be analysed in Section 2.7.3.

(iii) No lower bound on window size and congestion losses: This scenario, which we shall analyse in Section 2.7.4, is similar to the scenario with no lower bound except for the fact that when the window size reaches W_{max}, the source experiences a congestion loss with probability one. The upper bound in this scenario corresponds to the limitation due to the finite network capacity (i.e., $W_{recv} > W_{cong}$).

(iv) Lower and upper bounds on the window size: Finally, in Section 2.7.5 we shall analyse the scenario where both the lower and upper bounds are imposed. The recursive equation for W_n in this scenario is given by Equation (2.20). The other three scenarios are approximations to this scenario. Although the performance measures of interest can be obtained by analysing the exact scenario, it is generally not easy to obtain them in closed form. The approximations lead to some closed form expressions for the moments which give a better understanding of the system than the analysis of the exact scenario.

In the next section we shall relate the process $\{W_n\}$ in the first three scenarios with the workload evolution in a G/G/1 queue observed at particular epochs. This relation will enable us to obtain the distribution and the moments of W_n, thereby enabling us to obtain the throughput of the MIMD algorithm as a function of the loss rate, p.

2.7.1 Preliminary analysis

Consider the discrete time stochastic recursive equation[23]

$$W_{n+1} = \max(A_{n+1}W_n, 1). \tag{2.23}$$

The process $\{W_n\}$ can be viewed as a sequence of observations of a continuous time process sampled at certain, not necessarily equal, time intervals. The sequence $A_n \in (0, \infty)$ is assumed to be stationary and ergodic.

On taking the logarithm on both sides of Equation (2.23), we obtain

$$\log[W_{n+1}] = \max(\log[A_{n+1}] + \log[W_n], 0). \tag{2.24}$$

Using the substitutions $Y_n = \log[W_n]$ and $U_{n+1} = \log[A_{n+1}]$ in the above equation, we obtain

$$Y_{n+1} = \max(Y_n + U_{n+1}, 0). \tag{2.25}$$

The recursive equation Equation (2.25) is an additive recursive equation which has the same form as the recursive equation describing the amount of workload in a G/G/1 queue observed at, say, just before an arrival (see, for example, [Kle75]). The sequence U_{n+1} denotes the difference between the service time of the nth customer and the inter-arrival time between the nth and the $(n+1)$th customer. Since the introduced transformation, $\log(\cdot)$, is invertible, there is a one to one correspondence between the processes $\{Y_n, n \geq 0\}$ and $\{W_n, n \geq 0\}$. Following this observation, we can study the stability of the window size process $\{W_n, n \geq 0\}$ by studying the stability of $\{Y_n, n \geq 0\}$. Furthermore, the equivalence of the process $\{Y_n, n \geq 0\}$ and the workload in a $G/G/1$ queue will help us to obtain the steady state moments of W_n.

Theorem 2.7.1 *Assume that $E[\log A_0] < 0$. Then there exists a unique stationary ergodic process $\{W_n^*\}$, defined on the same probability space as $\{W_n\}$, that satisfies the recursion in Equation (2.23). Moreover, for any initial value $W_0 = w$, there is a random time T_w, which is finite with probability 1, such that $W_n = W_n^*$ for all $n \geq T_w$. If $E[\log A_0] > 0$ then W_n tends to infinity w.p.1 for any initial value $W_0 = w$.*

[23] Without loss of generality, we can assume $W_{min} = 1$. The results for any $W_{min} > 0$ can be obtained by scaling W_n appropriately.

Proof. According to Theorem 2A [GY95], if $\mathrm{E}[\log A_0] < 0$ then the stochastic process $\{Y_n\}$ converges to a stationary ergodic process $\{Y_n^*\}$ which is defined on the same probability space as $\{Y_n\}$ and is the unique stationary regime that satisfies Equation (2.25). This implies the statement for $W_n = \exp(Y_n)$. The last part of theorem similarly follows from [BB87, p. 36]. ■

Remark 2.7.1 *Due to Jensen's inequality and the concavity of the logarithm function,* $E[\log A_0] \leq \log E[A_0]$. *Hence,* $\log E[A_0] < 0$, *or equivalently* $E[A_0] < 1$, *is a sufficient condition for the stability of the window process* $\{W_n\}$ *(for the existence of a unique stationary ergodic regime and the convergence to this regime). However this condition is in general not a necessary one.*

Remark 2.7.2 *We stress the importance of the maximum operator in Equation (2.23). Indeed, if we eliminate it and write* $W_{n+1} = A_{n+1}W_n$ *then on taking the logarithm, we will obtain* $Y_{n+1} = \log[A_{n+1}] + Y_n$ *instead of Equation (2.25). Its solution is*

$$Y_n = Y_0 + \sum_{i=0}^{n-1} \log[A_{i+1}]. \tag{2.26}$$

Since $\{A_n\}$ *is stationary ergodic, the strong law of large numbers implies that if* $E\log[A_i] < 0$ *then* Y_n *will converge to* $-\infty$, *and, hence,* W_n *will converge to 0 which is clearly a bad estimate for the window size process. (If* $E\log[A_i] > 0$ *then* Y_n *and, hence,* W_n *will converge to* ∞ *which was also predicted by the model that took the minimum window into account.) Note that in the limiting case of* $E[\log A_i] = 0$, *if* A_i*'s are independent and identically distributed (i.i.d.) then* Y_n *is a null recurrent Markov chain and thus unstable.*

The logarithmic transformation allows us to obtain the moments of W_n in the stationary regime (i.e., moments of W_n^*) from the Laplace-Steiltjes Transform (LST) of Y_n in the stationary regime (i.e., LST of Y_n^*). The LST of Y_n^* is defined as

$$\mathbf{Y}(s) = \mathrm{E}[e^{-sY_n^*}], \tag{2.27}$$

for $s \in S$, where S is the region of convergence of $\mathbf{Y}(s)$. For a given integer $j \geq 0$, the jth moment of W_n^* is obtained as follows

$$\mathrm{E}[(W_n^*)^j] = \mathrm{E}[\exp(jY_n^*)] = \mathbf{Y}(-j), \tag{2.28}$$

where $-j$ is assumed to belong to S. If $-j \notin S$ then the corresponding moment is ∞. Thus, all finite moments of W_n^* can be obtained from the LST of Y_n^*.

A similar analysis can be done for the stochastic recursive equation of scenario *(ii)*,

$$W_{n+1} = \min(A_{n+1}W_n, W_{max}), \tag{2.29}$$

by making the transformation $Y_n = \log[W_{max}] - \log[W_n]$. The moments of W_n^* can then be obtained from the LST of Y_n^* using the relation

$$\mathrm{E}[(W_n^*)^j] = \mathrm{E}[W_{max}^j \exp(-jY_n^*)] = W_{max}^j \mathbf{Y}(k). \tag{2.30}$$

Since $\mathbf{Y}(s)$ is finite for $s \geq 0$, all the moments of W_n^* are finite.

The recursive equation for scenario (i), as given by Equation (2.21), is similar to Equation (2.23). Therefore, the analysis of this model can be done along the lines of the analysis of Equation (2.23). Similarly, the analysis of scenarios (ii) and (iii) can be done along the lines of the analysis of Equation (2.29). We note that the analysis of scenario (iii) is similar to that of scenario (ii). The equivalent queueing system of scenario (iii) can be obtained by deleting the idle periods of the equivalent queueing system of scenario (ii). The throughput of the MIMD algorithm, or the first moment of the window size, under different models, can be obtained from Equation (2.28) and Equation (2.30).

We note that the z-transform, which is defined for integer valued random variables, is a discrete analog of the LST. In the following sections, we shall derive the LST of the window size, W_n, for the three models.

In the rest of this subsection, we shall use the following notation. The process $\{W_n\}$ will denote the window size process. We shall denote by $\{Y_n\}$ the process obtained upon transforming $\{W_n\}$. The process $\{W_n\}$ converges in the limit $n \to \infty$ to a stationary ergodic process which we shall denote by W. Similarly, Y shall denote the stationary ergodic process to which $\{Y_n\}$ converges in the limit $n \to \infty$. We shall denote the probability distribution functions of Y_n and Y by π_n and π, respectively.

2.7.2 Scenario (i): no upper bound on window size

In this subsection, we analyse scenario (i) in which the recursive equation for the window size embedded at intervals of RTT is

$$W_{n+1} = \max(A_{n+1}W_n, W_{min}). \tag{2.31}$$

The sequence $\{A_n, n \geq 1\}$ is i.i.d. such that

$$A_n = \begin{cases} \alpha & \text{w.p.} \quad 1-p; \\ \beta & \text{w.p.} \quad p. \end{cases} \tag{2.32}$$

Along the lines of the preliminary analysis, we make the transformation

$$Y_n = \frac{\log[W_n] - \log[W_{min}]}{\log[\alpha]}, \tag{2.33}$$

in which the normalization by $\log[\alpha]$ is made for convenience. The recursive equation for the process $\{Y_n\}$ is given by (from Equation (2.25))

$$Y_{n+1} = \begin{cases} Y_n + 1 & \text{w.p.} \quad 1-p; \\ (Y_n - k)^+ & \text{w.p.} \quad p. \end{cases} \tag{2.34}$$

where $k = -\frac{\log[\beta]}{\log[\alpha]}$. Since $\beta < 1$, the number k is positive. For the process $\{Y_n\}$ to be positive recurrent, the necessary and sufficient condition is $\mathrm{E}\log[A_n] < 0$, or, equivalently,

$$(k+1)p > 1. \tag{2.35}$$

The above inequality gives the condition on p for which scenario (i) is a valid approximation to the true window behaviour. In the rest of this subsection, we shall assume that the above condition is satisfied. We also make the following two assumptions.

Assumption 2.7.1 $k = -\frac{\log[\beta]}{\log[\alpha]}$ *is an integer.*

Assumption 2.7.2 $\frac{\log[W_{min}]}{\log[\alpha]}$ *is an integer.*

With these two assumptions, the state space of Y_n reduces to the set of non-negative integers, $\mathbb{Z}_+ = \{0, 1, 2, ...\}$. Thus, Y_n can be modelled as a discrete state space Markov chain. Using the inverse transformation of Equation (2.33), the state $Y_n = i$ maps to $W_n = W_{min}\alpha^i$. The state transition diagram of this scenario is shown in Figure 2.4.

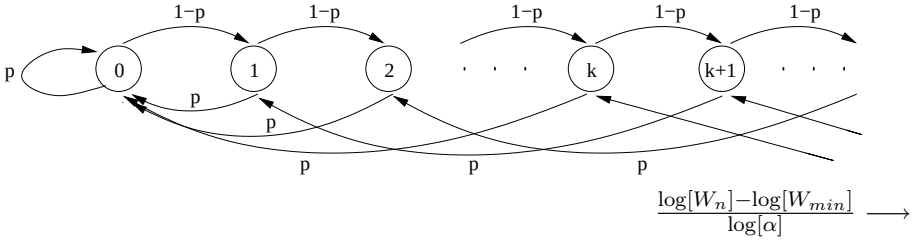


Figure 2.4: State transition diagram of $Y_n = \frac{\log[W_n]-\log[W_{min}]}{\log[\alpha]}$.

Let $\pi_n(j), j \in \mathbb{Z}_+$, be the probability of Y_n being in state j. From the forward equation for the probabilities of a Markov chain,

$$\pi_{n+1}(j) = \begin{cases} (1-p)\pi_n(j-1) + p\pi_n(j+k), & j \geq 1; \\ p\sum_{i=0}^{k} \pi_n(i), & j = 0. \end{cases} \tag{2.36}$$

Let $\mathbf{Y}_n(z)$ denote the z-transform of Y_n. The function $\mathbf{Y}_n(z)$ is defined as

$$\mathbf{Y}_n(z) = \sum_{j=0}^{\infty} \pi_n(j) z^j. \tag{2.37}$$

From Equation (2.36) and Equation (2.37), we obtain

$$\begin{aligned} \mathbf{Y}_{n+1}(z) - \pi_{n+1}(0) &= \sum_{j=1}^{\infty} \pi_{n+1}(j) z^j \\ &= (1-p)\sum_{j=1}^{\infty} \pi_n(j-1) z^j + p\sum_{j=1}^{\infty} \pi_n(j+k) z^j. \end{aligned} \tag{2.38}$$

Let $\mathbf{Y}(z) = \lim_{n\to\infty} \mathbf{Y}_n(z)$. Let W denote the stationary process to which $\{W_n\}$ converges in the limit $n \to \infty$. We note that the stationary process of $\{Y_n\}$, Y, and W are related as $W = W_{min}\alpha^Y$. Therefore, the jth moment of W is given by

$$\mathrm{E}[W^j] = \mathrm{E}[(W_{min}\alpha^Y)^j] = W_{min}^j \mathbf{Y}(\alpha^j). \tag{2.39}$$

Theorem 2.7.2 *(a)* $\mathbf{Y}(z)$ *is given by*

$$\mathbf{Y}(z) = \frac{1 - 1/z_0}{1 - z/z_0}, \tag{2.40}$$

where z_0 *is the unique root of*

$$\frac{(1-p)}{p} z^{k+1} - \frac{1}{p} z^k + 1 = 0 \tag{2.41}$$

that lies outside the closed unit disc.
(b)

$$Prob(W \geq w) = \left(\frac{w}{W_{min}} \right)^{-\log[z_0]/\log[\alpha]}. \tag{2.42}$$

Proof. From Theorem 2.7.1 we can conclude that the Markov chain Y_n is stationary and ergodic. In particular, this implies that $\mathbf{Y}_n$ converges to $\mathbf{Y}$ and that $\pi_n(\cdot)$ converges to $\pi(\cdot)$, the stationary probability distribution function of Y.

To prove part (a), from Equation (2.38) we obtain

$$\begin{aligned}
\mathbf{Y}(z) - \pi(0) &= (1-p)z\mathbf{Y}(z) + pz^{-k} \sum_{j=0}^{\infty} \pi_n(j+1+k) z^{j+1+k} \\
\mathbf{Y}(z)(1-(1-p)z) &= \pi(0) + pz^{-k} \sum_{j=0}^{\infty} \pi_n(j+1+k) z^{j+1+k} \\
&= \pi(0) + pz^{-k}(\mathbf{Y}(z) - \sum_{i=0}^{k} \pi(i) z^i),
\end{aligned}$$

and hence,

$$\mathbf{Y}(z)((1-(1-p)z)z^k - p) = z^k \pi(0) - p \sum_{i=0}^{k} \pi(i) z^i.$$

Since $\pi(0) = p \sum_{i=0}^{k} \pi_n(i)$, $\mathbf{Y}(z)$ can be expressed as

$$\mathbf{Y}(z) = \frac{\sum_{i=0}^{k-1} \pi(i)(z^k - z^i)}{-\frac{(1-p)}{p} z^{k+1} + \frac{1}{p} z^k - 1}. \tag{2.43}$$

Under the stability condition of Equation (2.35), $\mathbf{Y}(z)$ exists and is analytic in the open disc $\{z : |z| < 1\}$. The numerator of Equation (2.43) has at most $k-1$ zeros inside the unit circle and one zero on the unit circle. Hence, there can be at most $k-1$ zeros of the denominator of Equation (2.43) within the unit circle as any more zeros will make $\mathbf{Y}(z)$ non-analytic. Using Rouche's theorem [Kra99] we can show that there are at least k zeros of the denominator inside and on the unit circle. As $z = 1$ is a zero of the denominator, there are at least $k-1$ zeros inside the unit circle. From the two previous arguments, there are exactly $k-1$ zeros of the denominator within the unit circle and they must be the same as those of the numerator for $\mathbf{Y}(z)$ to be analytic [Kle75]. Hence, $\mathbf{Y}(z)$ reduces to Equation (2.40).

To prove part (b), we note that the distribution of Y can be obtained by inverting $\mathbf{Y}(z)$, and is given by

$$\pi(j) = (1 - 1/z_0)(1/z_0)^j, \quad j \geq 0. \tag{2.44}$$

The above equation, together with the relation $W = W_{min}\alpha^Y$, gives Equation (2.42). ■

Corollary 1 *Let* $\zeta = \frac{\log[z_0]}{\log[\alpha]}$. *The* j*th moment of* W *is given by*

$$E[W^j] = \begin{cases} W_{min}\frac{\zeta}{\zeta - j} & j < \zeta; \\ \infty & j \geq \zeta. \end{cases} \tag{2.45}$$

This follows from Equation (2.39) and Equation (2.40). The z-transform $\mathbf{Y}(z)$ is analytic for $z < z_0$. Hence, the jth moment of W is finite if $j < \zeta$. The window size distribution can be seen to become heavy tailed for $\zeta \leq 2$. Thus, for a given loss rate, p, either α or β can be suitably chosen in order to reduce the variance of the window size.

With this we have obtained the distribution of the window size process, and its moments when the loss rate is quite high (i.e., $p > 1/(k+1)$. Next, we study the complementary scenario where the loss rate is small (i.e., $p < 1/(k+1)$).

2.7.3 Scenario (ii): no lower bound on window size

In this subsection, we consider the scenario where the window at the sender is limited to W_{max}. The upper bound is due to the limitation imposed by the receiver's advertised window. We note that losses are only due to random events other than congestion. We transform the process $\{W_n\}$ in the following way,

$$Y_n = \frac{\log[W_{max}] - \log[W_n]}{\log[\alpha]}. \tag{2.46}$$

We make the following assumption along with Assumption 2.7.2.

Assumption 2.7.3 $\frac{\log[W_{max}]}{\log[\alpha]}$ *is an integer.*

The state transition diagrams of $\frac{\log[W_n]}{\log[\alpha]}$ and of Y_n are shown in Figure 2.5. From the transformation in Equation (2.46), the state $Y_n = i$ maps to the state $W_n = W_{max}\alpha^{-i}$. The process $\{Y_n\}$ satisfies the recursive equation

$$Y_{n+1} = \begin{cases} Y_n + k & \text{w.p.} \quad p; \\ (Y_n - 1)^+ & \text{w.p.} \quad 1 - p. \end{cases} \tag{2.47}$$

From Theorem 2.7.1, for the process $\{Y_n\}$ to be positive recurrent, the necessary and sufficient condition is

$$(k+1)p < 1. \tag{2.48}$$

In the rest of this section, we assume that the above condition is satisfied.

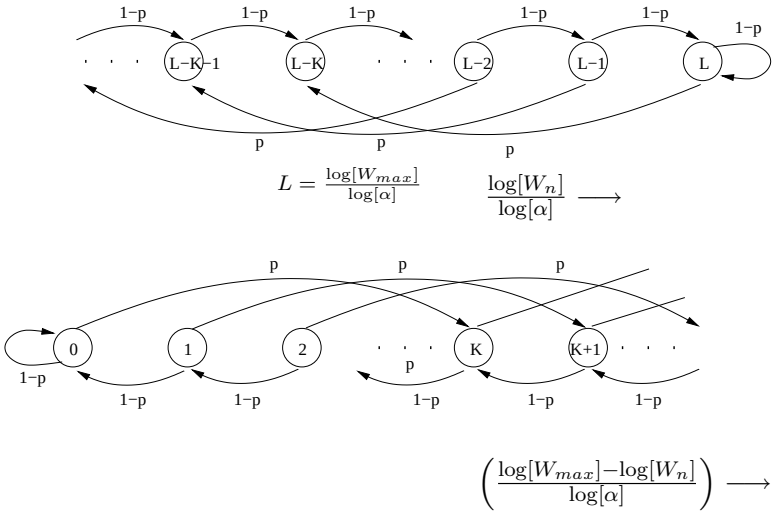


Figure 2.5: State transition diagram of $\frac{\log[W_n]}{\log[\alpha]}$ (top figure) and $Y_n = \frac{\log[W_{max}]-\log[W_n]}{\log[\alpha]}$ (bottom figure) with no congestion losses.

As in the previous subsection, let Y denote the stationary process to which $\{Y_n\}$ converges in the limit $n \to \infty$. The balance equations for computing the steady state probabilities are given by

$$\begin{aligned}
\pi(0) &= \frac{(1-p)}{p}\pi(1), \\
\pi(i) &= (1-p)\pi(i+1), \quad i = 1, ..., k-1. \\
\pi(i) &= p\pi(i-k) + (1-p)\pi(i+1), \quad i \geq k.
\end{aligned}$$

These equations are similar to those for the number of customer in a bulk arrival queue.

Theorem 2.7.3 *The z-transform of Y, $\mathbf{Y}(z)$, is given by*

$$\mathbf{Y}(z) = (1-(k+1)p)\frac{1-z}{pz^{k+1} - z + (1-p)}. \tag{2.49}$$

Proof. Following similar arguments from Kleinrock [Kle75], we can write the z-transform as follows.

$$\sum_{i=1}^{\infty} \pi(i)z^i = \sum_{i=1}^{\infty} p\pi(i-k)z^i + \sum_{i=1}^{\infty}(1-p)\pi(i+1)z^i,$$

where $\pi(i-k) = 0$ for $i < k$. Therefore,

$$\begin{aligned}
\mathbf{Y}(z) - \pi(0) &= pz^k \sum_{i=1}^{\infty} \pi(i-k)z^{i-k} + z^{-1}\sum_{i=1}^{\infty}(1-p)\pi(i+1)z^{i+1} \\
&= pz^k\mathbf{Y}(z) + z^{-1}(1-p)(\mathbf{Y}(z) - z\pi(1) - \pi(0)),
\end{aligned}$$

which gives

$$\mathbf{Y}(z)(z - pz^{k+1} - (1-p)) = \pi(0)(z - (1-p) - zp)). \tag{2.50}$$

Using the equality $\mathbf{Y}(1) = 1$ along with L'Hôpital's rule, we get $\pi(0) = [1-(k+1)p]/[1-p]$. Hence, we obtain $\mathbf{Y}(z)$ as given by Equation (2.49). ■

Corollary 2 *The moments of* $W = W_{max}\alpha^{-Y}$ *are given by*

$$E[W^j] = E[(W_{max}\alpha^{-Y})^j)] = W_{max}^j E[\alpha^{-jY}] = W_{max}^j \mathbf{Y}(\alpha^{-j}). \tag{2.51}$$

The distribution of Y can be found by inverting $\mathbf{Y}(z)$ using partial fraction expansion. The distribution can be seen to be a weighted sum of geometric distributions.

2.7.4 Scenario (iii): no lower bound on window size and congestion losses

In the present scenario, along with random losses, a loss occurs when the window size reaches W_{max}. The upper bound is due to the limitation of the network capacity. When this sending rate is exceeded congestion sets in, and the source experiences a loss.

We make use of the same transformation that was made in Scenario (ii). The state transition diagram of Y_n is shown in Figure 2.6. As in Scenario (ii), the state $Y_n = i$ maps to the state $W_n = W_{max}\alpha^{-i}$. The only difference in transitions from the previous scenario is at state $Y = 0$. In

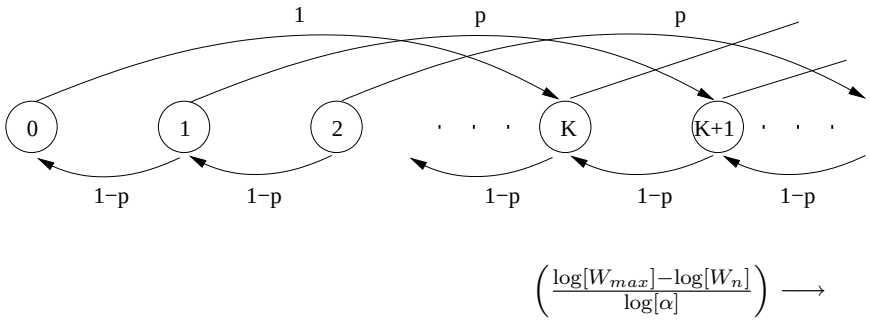


Figure 2.6: State transition diagram of $Y_n = \frac{\log[W_{max}]-\log[W_n]}{\log[\alpha]}$ with congestion losses.

this scenario, there is a jump with probability one from state 0 to state k. The balance equations for the steady state probability distribution of Y are given by

$$\begin{aligned} \pi(i) &= (1-p)\pi(i+1), \quad i = 0, ..., k-1, \\ \pi(i) &= p\pi(i-k) + (1-p)\pi(i+1) + (1-p)\pi(0)\delta_{i-k}, \quad i \geq k. \end{aligned}$$

Theorem 2.7.4 *The z-transform of* Y, $\mathbf{Y}(z)$, *is given by*

$$\mathbf{Y}(z) = \left(\frac{1-(k+1)p}{(k+1)}\right) \frac{1-z^{k+1}}{pz^{k+1} - z + (1-p)}. \tag{2.52}$$

Proof. We can find $\mathbf{Y}(z)$ as follows.

$$\sum_{i=1}^{\infty} \pi(i) z^i = pz^k \sum_{i=1}^{\infty} \pi(i-k) z^{i-k} + \frac{1}{z} \sum_{i=1}^{\infty} (1-p)\pi(i+1) z^{i+1} + z^k (1-p)\pi(0), \tag{2.53}$$

which implies

$$\mathbf{Y}(z) - \pi(0) = pz^k\mathbf{Y}(z) + z^{-1}(1-p)(\mathbf{Y}(z) - z\pi(1) - \pi(0)) + z^k(1-p)\pi(0), \tag{2.54}$$

and gives the relation

$$\mathbf{Y}(z)(z - pz^{k+1} - (1-p)) = \pi(0)(1-p)(z^{k+1} - 1). \tag{2.55}$$

Using the equality $\mathbf{Y}(1) = 1$ along with L'Hôpital's rule, we get $\pi(0) = [1-(k+1)p]/[(k+1)(1-p)]$. Hence, we obtain $\mathbf{Y}(z)$ as given by Equation (2.52). ■

As for Scenario (ii), we can obtain the distribution of Y, and hence that of W, by inverting the z-transform. We can also obtain the moments of W directly from $\mathbf{Y}(z)$ using the relation

$$\mathrm{E}[W^j] = W_{max}^j \mathbf{Y}(\alpha^{-j}). \tag{2.56}$$

2.7.5 Scenario (iv): lower and upper bounds on the window size

The scenarios analysed in the previous three subsections assumed either a lower or an upper bound on the congestion window. Although approximate, the performance measures obtained through the analysis of scenarios $(i) - (iii)$ are easy to evaluate and provide simple expressions to obtain the throughput of the connection. In this subsection, we present an analysis when the congestion window is bounded from above and from below. The recursive equation for this scenario is

$$W_{n+1} = \max(\min(A_{n+1}W_n, W_{max}), W_{min}). \tag{2.57}$$

We make Assumptions 2.7.1, 2.7.2, and 2.7.3.

Let $Y_n = \frac{\log[W_{max}] - \log[W_n]}{\log[\alpha]}$ be the transformation applied to W_n. Let $L = \frac{\log[W_{max}] - \log[W_{min}]}{\log[\alpha]}$ be the number of states of Y_n. The recursive equation for the process $\{Y_n\}$ is given by

$$Y_{n+1} = \begin{cases} \min(Y_n + k, L) & \text{w.p.} \quad p, \\ \max(Y_n - 1, 0) & \text{w.p.} \quad 1-p. \end{cases} \tag{2.58}$$

The state transition diagram of Y_n is shown in Figure 2.7. The Markov chain Y_n has a finite

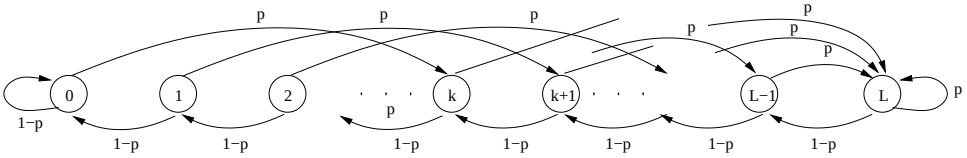


Figure 2.7: State transition diagram of Y_n.

state space, and is a communicating Markov chain with $L+1$ states. Therefore, its steady state probabilities exist for $0 < p < 1$. The steady state probability distribution of Y satisfies the balance equations

$$\begin{aligned} \pi(m) &= (1-p)\pi(m+1) + p\pi(m-k), \quad m = 1, 2, ..., L-1, \\ \pi(0) &= \frac{1-p}{p}\pi(1). \end{aligned} \tag{2.59}$$

where we define $\pi(m) = 0$ for $m < 0$.

Proposition 2.7.1 *Let $m = r(k+1)+j$, where $j = m \mod (k+1)$. The steady state probability $\pi(m)$ is given by*

$$\pi(m) = \pi(0)\frac{p}{(1-p)^m}\cdot\sum_{i=0}^{r}(-1)^i\left(p^{i-1}C_{i-1,(r-i)(k+1)+j}+p^iC_{i,(r-i)(k+1)+j-1}\right)(1-p)^{ik}, \quad m = 0,1,2,...,L. \tag{2.60}$$

The probability $\pi(0)$ can be obtained using the normalizing condition $\sum_{m=0}^{L}\pi(m) = 1$. The coefficients, $C_{i,j}$, are given by

$$C_{i,j} = C_{j,i} = \binom{i+j}{i} = \binom{i+j}{j}. \tag{2.61}$$

$C_{i,j}$ can also be calculated from the recursion

$$C_{i,j} = C_{i,j-1} + C_{i-1,j}. \tag{2.62}$$

The first few values of $C_{i,j}$ are given in the array below

(i,j)	-1	0	1	2	3	...
-1	0	1	0	0	0	...
0	0	1	1	1	1	...
1	0	1	2	3	4	...
2	0	1	3	6	10	...
3	0	1	4	10	20	...
	⋮	⋮	⋮	⋮	⋮	⋱

(2.63)

Proof. It can be shown using induction that the steady state probabilities given by Equation (2.60) satisfy the steady state balance equations in Equation (2.59). The complete proof is given in Appendix A.1. ■

The steady state probabilities of the window size, $\text{Prob}(W = w)$, can be obtained by using the relation

$$\text{Prob}(W = w) = \text{Prob}\left(Y = \frac{\log[W_{max}] - \log[w]}{\log[\alpha]}\right). \tag{2.64}$$

We note that $\frac{\log[W_{max}]-\log[w]}{\log[\alpha]}$ is an integer since the state space of W, $\mathcal{W}$, is of the form $\mathcal{W} = \{W_{min}\cdot\alpha^i, i = 0,1,...,L\}$. The jth moment of W can be obtained from

$$\text{E}[W^j] = \sum_{w\in\mathcal{W}} w^j\text{Prob}(W = w). \tag{2.65}$$

The computational cost for obtaining the moments using Equation (2.65) is much higher than if we use either Equation (2.45) or Equation (2.51).

In a similar way, the steady state probabilities for the stationary process W can be derived for the scenario when there are lower and upper bounds on the window size and there are congestion

losses at W_{max} (i.e., Scenario (iii) with a lower bound on the window size). For this scenario, the recursion of Equation (2.59) is modified to

$$\begin{aligned} \pi(m) &= (1-p)\pi(m+1) + p\pi(m-k), \quad m = k+1, k+2, ..., L-1, \\ \pi(i) &= (1-p)^{k-i}\pi(k), \quad i = 0, 1, ..., k. \end{aligned} \tag{2.66}$$

It can be seen that Equation (2.66) for $m \geq k+1$ is same as Equation (2.59) for $m \geq 1$. Therefore, we can use Proposition 2.7.1 to obtain the probabilities $\pi(m), m \geq k+1$ as a function of $\pi(k)$. Since the probabilities are obtained as a function of $\pi(k)$, we first substitute $\pi(0) = \pi(k)\frac{(1-p)}{p}$ in Equation (2.60), and then we obtain

Proposition 2.7.2 *The steady state probability $\pi(m)$ is given by*

$$\begin{aligned} \pi(m) &= \pi(k)\frac{1}{(1-p)^{m-k}} \cdot \sum_{i=0}^{r}(-1)^i \big(p^{i-1}C_{i-1,(r-i)(k+1)+j} + p^i C_{i,(r-i)(k+1)+j-1}\big)(1-p)^{ik}, \\ & \qquad m = k+1, k+2, ..., L, \\ \pi(m) &= \pi(k)(1-p)^{k-i}, \quad m = 0, 1, ..., k. \end{aligned}$$

The probability $\pi(k)$ can be obtained from the normalizing condition $\sum_{m=0}^{L}\pi(m) = 1$.

We note that in the above equations we compute the probabilities with respect to $\pi(k)$ unlike in Equation (2.60) where the probabilities are computed relative to $\pi(0)$.

Thus, we have obtained the steady state distribution of the window size of the MIMD congestion control algorithm in the presence of a *window-independent loss* process. We shall compare the results obtained in this subsection with simulations in Section 2.10. Next, we study the model with a *window-dependent loss* process.

2.8 Window-dependent loss process

In this section, we consider a model in which the losses in an RTT depend on the window size in that RTT. Specifically, we assume that each packet is lost (or, equivalently, is in error) with a constant probability q. As a consequence of this assumption, the probability of one or more packet losses in an RTT is no longer independent of the window size in that RTT. First, we present the model with the *window-dependent loss* process. Then we propose an approximation to this model which will enable us to compute the average window size in this model using the expression for the average window size in the *window-independent loss* model with no lower bound (Scenario (ii)).

Let W_n be the window size at the end of the nth RTT. Therefore, the source transmitted W_n packets during the nth RTT. Let p_n denote the probability that one or more losses occurred during the nth RTT. Then,

$$p_n = 1 - (1-q)^{W_n}. \tag{2.67}$$

The source will receive the loss notifications for these W_n packets during the $(n+1)$th RTT, and will react accordingly. The window size W_{n+1} satisfies the recursive equation

$$W_{n+1} = \max\min(A_{n+1}W_n, W_{max}), W_{min}), \tag{2.68}$$

where A_{n+1} is defined as

$$A_{n+1} = \begin{cases} \alpha & \text{w.p.} \quad 1-p_n, \\ \beta & \text{w.p.} \quad p_n. \end{cases} \tag{2.69}$$

Instead of obtaining an explicit expression for the distribution function of the steady state process W, we propose an approximation through which we can obtain a closed form expression for the average sending rate.

Let us assume that $W_{min} = 0$. This model was studied in Scenario (ii), albeit with a *window-independent loss* process. For $q \cdot E[W] < 1$ [24] , we can approximate p_n as

$$p_n \approx qW_n. \tag{2.70}$$

Using the approximation of Equation (2.70), the average loss rate, $\mathrm{E}[p]$, in an RTT is given by

$$\mathrm{E}[p] = q\mathrm{E}[W]. \tag{2.71}$$

We now substitute $\mathrm{E}[p] = q\mathrm{E}[W]$ for the probability of loss in the expression for computing the average window size in Scenario (ii) (i.e., in Equation (2.51) with $j = 1$.). For $j = 1$, Equation (2.51) gives us

$$\mathrm{E}[W] = W_{max}(1-(k+1)p)\frac{1-\alpha^{-1}}{p\alpha^{-(k+1)} - \alpha^{-1} + (1-p)}, \tag{2.72}$$

and Equation (2.72) together with Equation (2.71) gives

$$\mathrm{E}[W] = W_{max}(1-(k+1)q\mathrm{E}[W])\frac{1-\alpha^{-1}}{q\mathrm{E}[W]\alpha^{-(k+1)} - \alpha^{-1} + (1-q\mathrm{E}[W])}. \tag{2.73}$$

Equation (2.73) is a quadratic equation in $\mathrm{E}[W]$, namely

$$c_2\mathrm{E}[W]^2 + c_1\mathrm{E}[W] + c_0 = 0, \tag{2.74}$$

where $c_2 = q\frac{1-\alpha^{-(k+1)}}{1-\alpha^{-1}}$, $c_1 = -(1+(k+1)qW_{max})$, and $c_0 = W_{max}$. Therefore, its roots can be explicitly written as

$$\mathrm{E}[W]_{1,2} = \frac{-c_1 \pm \sqrt{c_1^2 - 4c_2c_0}}{2c_2}. \tag{2.75}$$

Proposition 2.8.1 *The solution of Equation (2.74) which satisfies the inequality* $E[W] \leq W_{max}$ *is*

$$E[W] = \frac{-c_1 - \sqrt{c_1^2 - 4c_2c_0}}{2c_2}. \tag{2.76}$$

[24] In some way, this is a self referential condition. That is to say, even though we are not aware of the value of $\mathrm{E}[W]$, we assume this condition is satisfies in the computation of $\mathrm{E}[W]$. Later, we shall see that the $\mathrm{E}[W]$ obtained does indeed satisfy this condition, i.e, this condition is self consistent. Also, we can replace this condition with $qW_{max} < 1$, but this condition turns out to be restrictive on q.

Proof. Please see Appendix A.2. ■

The average window size obtained in Equation (2.76) is a function of W_{max}. For networks with sufficiently large capacity (i.e., when $W_{max} \to \infty$)), the dependence of the average window size on the packet loss probability q is given by the following proposition.

Proposition 2.8.2 *In large capacity networks, the average window size observed by an MIMD connection in the presence of a packet loss probability q is given by*[25]

$$\lim_{W_{max}\to\infty} E[W] = \frac{1}{q(k+1)}. \tag{2.77}$$

Proof. Please see Appendix A.3. ■

Therefore, the throughput obtained by an MIMD connection is inversely proportional to the packet loss probability, q. This result is similar to the result obtained in [BB01] and [Kel03b]. We shall also obtain a similar proportionality in Chapter 3. We can contrast this inverse proportionality of the throughput with the inverse square root proportionality (i.e., $\propto q^{-0.5}$) of the throughput for an AIMD connection [OKM96].

In Equation (2.76) we have obtained an expression for the approximate average window size in the model with *window-dependent loss* process. We shall compare this approximation with simulation results in Section 2.10.

2.9 Window-independent first-order Markovian loss process

In this section, we shall assume the loss rates process to be a first-order Markov chain. The first-order Markov model introduces correlations between the loss rates in successive RTTs. We shall assume that the lower bound on the window size, W_{min}, is zero. This model is similar to Scenario (ii), albeit with an first-order Markov loss process.

Let R_n denote the state of the loss process in the nth round trip time. We assume that R_n can take two values:"GOOD" (state 0) and "BAD" (state 1). When R_n is in the "GOOD" state, the loss rate is zero. When R_n is in the "BAD" state, the loss rate is unity. Let The transitions of R_n take place at intervals of RTT. Let P denote the transition probability matrix for R_n. We define P as

$$P = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & \quad 1 \end{array} \\ \begin{pmatrix} g & 1-g \\ 1-b & b \end{pmatrix} \end{array}. \tag{2.78}$$

We define the vector $\mathbf{1}' = [1 \quad 1]'$, i.e., the transpose of the vector $[1 \quad 1]$. Let ξ denote the steady state probability vector of the loss process. Then, ξ satisfies the equations

$$\xi = \xi P \ \text{ and } \ \xi \mathbf{1}' = 1, \tag{2.79}$$

and is given by

$$\xi = \left[\frac{1-b}{2-b-g} \quad \frac{1-g}{2-b-g} \right]. \tag{2.80}$$

[25] From this relation, we can see that the condition $q\mathrm{E}[W] < 1$ is indeed satisfied.

The average loss rate observed by the source can be computed as

$$p_a = \frac{1-g}{2-b-g} \tag{2.81}$$

With this definition of the loss process, we note that b is the probability of one or more losses in the $(n+1)$th RTT given there were one or more losses in the nth RTT. We shall call b as the burstiness parameter. For a fixed average loss rate, we shall be interested in evaluating the effect of the burstiness parameter on the expected window size.

The process W_n satisfies the recursive equation

$$W_{n+1} = \begin{cases} \alpha W_n, & \text{if} \quad R_n = 0, \\ \beta W_n, & \text{if} \quad R_n = 1. \end{cases} \tag{2.82}$$

As in Section 2.7.3, we define Y_n as

$$Y_n = \frac{\log[W_{max}] - \log[W_n]}{\log[\alpha]}. \tag{2.83}$$

The couple (Y_n, R_n) forms a two-dimensional Markov chain with a transition probability matrix of (Y_n, R_n) which is of $M/G/1$-type [Neu81]. The necessary and sufficient condition for the process (Y_n, R_n) to converge to a stationary and ergodic process (Y, R) is given by [Neu81]

$$p_a(k+1) < 1. \tag{2.84}$$

Let $\mathbf{Y}_0(z)$ be defined as

$$\mathbf{Y}_0(z) = \sum_j \text{Prob}(Y = j, R = 0) z^j. \tag{2.85}$$

We define $\mathbf{Y}_1(z)$ in a similar way.

Proposition 2.9.1 *The vector* $\mathbf{Y}(z) = [\mathbf{Y}_0(z) \quad \mathbf{Y}_1(z)]$ *can be derived as*

$$\mathbf{Y}(z)[QPz^{k+1} - Iz + (I-Q)P] = (1-z)\mathbf{Y}(0)(I-Q)P, \tag{2.86}$$

where

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.87}$$

and I *is the* 2×2 *identity matrix.*

Proof. Since the matrix P is of rank one, from Theorem 7 in [LZ96], the vector $\mathbf{Y}(0)$ can be obtained up to a multiplicative constant by solving

$$\mathbf{Y}(0) = \mathbf{Y}(0)(PQ + (I-P)QG), \tag{2.88}$$

where G is a rank one matrix given by $G = \mathbf{1}'[g \quad 1-g]$. The multiplicative constant can be computed using the normalization equation $\mathbf{Y}(1)\mathbf{1}' = \mathbf{1}$ along with Equation (2.86). Hence, we get

$$\mathbf{Y}(z) = (1-z)(1-(k+1)p_a)\begin{bmatrix} 1 & \frac{1-g}{g} \end{bmatrix}(I-Q)P(QPz^{k+1} - Iz + (I-Q)P)^{-1}. \tag{2.89}$$

■

Let $\mathrm{E}[W_0]$ and $\mathrm{E}[W_1]$ denote the expected window size when the loss process is in state 0 and in state 1, respectively. Since W and Y and related by Equation (2.83), the vector $\mathrm{E}[\mathbf{W}^j] = [\mathrm{E}[W_0^j] \quad \mathrm{E}[W_1^j]]$ can be obtained as

$$\mathrm{E}[\mathbf{W}^j] = W_{max}^j \mathbf{Y}(\alpha^{-j}). \tag{2.90}$$

2.10 Simulation results

In this section, we shall compare the results obtained through analysis with simulation results.

2.10.1 Simulation setup

The simulation were performed using the publicly available software *ns-2* [MF]. As mentioned in Section 2.3, Scalable TCP uses the MIMD algorithm in the *congestion avoidance phase*. Therefore, we shall validate our analytical findings on the MIMD algorithm by simulation results of Scalable TCP.

The simulation setup consists of a source which is connected to a destination by means of a data link. The link bandwidth is 150Mbps and the two way propagation delay is 120ms. The source has infinite amount of data to send. Since Scalable TCP differs from the standard TCP only in the increase algorithm, the rest of the congestion control mechanism (i.e., the i.e., the *fast recovery/retransmit phase*, timers, etc.) of Scalable TCP is taken to be the same as the NewReno version of the standard TCP. The bandwidth-delay product for this system is approximately 2250 packets (packet size is 1040 bytes). In the Scalable TCP algorithm that we have implemented in *ns-2*, the following assumptions are made:

1. The minimum window size, W_{min}, is 8. It has been recommended in [Kel03b] to use the Scalable algorithm after a certain threshold. The author proposed $W_{min} = 16$.

2. There is no separate slow start phase since slow start can be viewed as a multiplicative increase algorithm with $\alpha = 2$.

3. The increase factor, α, is taken to be 1.01, and the decrease factor, β, is taken to be 0.86. This value of β gives $k = -\frac{\log[\beta]}{\log[\alpha]} \approx 15$. The values of α and β recommended in [Kel03b] are $\alpha = 1.01$ and $\beta = 0.875$.

In the simulations, the performance measures of interest are obtained by sampling the window size at intervals of RTT = 0.12s. We would like to note that the value of the RTT is very close to the propagation delay in the present setting, hence it does not vary much. Our main performance measure of interest will be the throughput obtained by the source. We define throughput as the average number of packets sent per RTT. With this definition, the throughput from the analysis can be obtained by computing $\mathrm{E}[W]/\mathrm{RTT}$.

2.10.2 Window-independent loss process

First, we discuss the results of the simulations for the *window independent loss* process, which was analysed in Section 2.7.

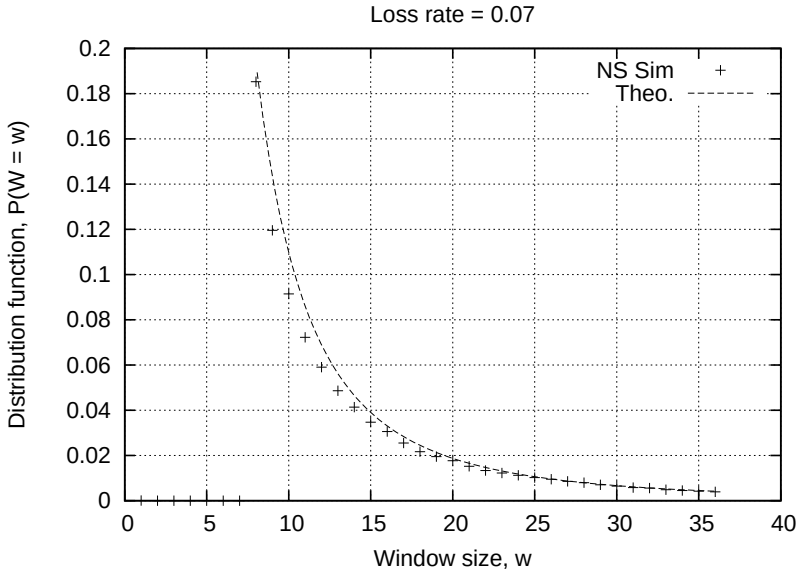


Figure 2.8: Distribution function of the window size, W, for Scenario (i). $\zeta = 1.55$.

1. *Scenario (i):* In Figures 2.8 and 2.9, the probability distribution function of W, which obtained through analysis (obtained from Equation (2.42)) and simulations, is shown for two different values of the loss rate, p. Depending on the value of the root, z_0, of Equation (2.41), the distribution can be seen to become heavy tailed. For example, for $p = 0.07$, the tail decreases at rate 1.55 indicating the heavy tailed nature of the window size. Figure 2.10 shows the throughput as a function of the loss rate, p. The error bars are the 99% confidence intervals. In the models which we considered, the window size could take non-integer values. In practice, however, the window size (or, strictly speaking, the number of packets in the network) takes only integer values. For example, when the window size is 8.5, the sender sends 8 packets. This can result in a discrepancy between the simulations and the theoretical function.

2. *Scenario (ii):* Figure 2.11 shows the throughput as a function of the loss rate, p, for the scenario in which the maximum window at the sender is limited by the receiver's advertised window. The receiver buffer is assumed to be limited to 500 packets. The error bars are the 99% confidence intervals. A good match is observed between the simulations and the analysis.

3. *Scenario (iv):* We now compare the results of Scenario *(i)* and Scenario *(ii)* with those of the Scenario *(iv)* and simulations. Since Scenarios *(i)* and *(ii)* are approximations to Scenario *(iv)*, we shall be able to discern the range of loss rates for which the approximations are fairly accurate. In Figure 2.12, the throughput obtained from the different scenarios and simulations is plotted as a function of loss rate. The vertical line $p = 1/(k+1)$ separates the two regions where Scenario (i) and Scenario (ii) are valid, respectively. As p approaches $1/(k+1)$ from either direction, the approximations diverge from the simulation results. However, Scenario (i) gives a good estimate when $(k+1)p >> 1$, i.e., $p >> 0.625$ ($k = 15$ in the simulations). Similarly, Scenario (ii) gives a good approximation of the system when $p << 0.625$. The

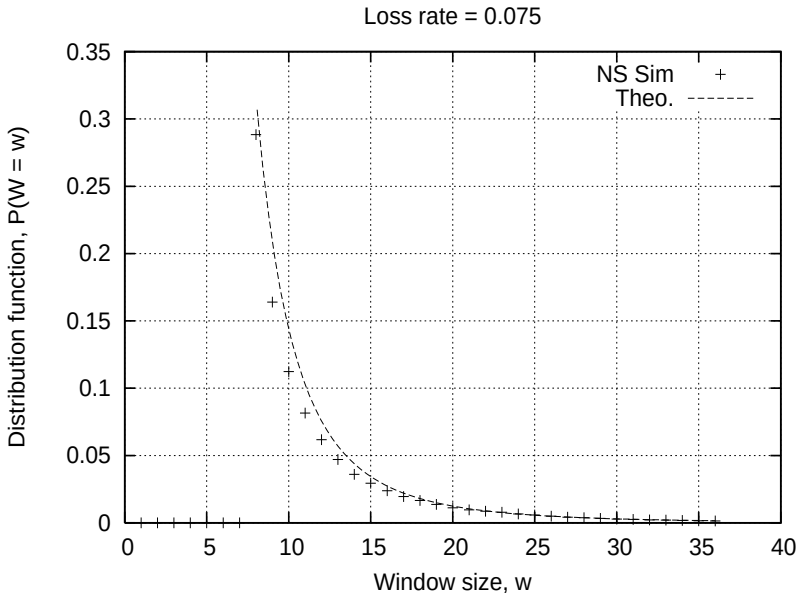


Figure 2.9: Distribution function of the window size, W, for Scenario (i). $\zeta = 2.53$.

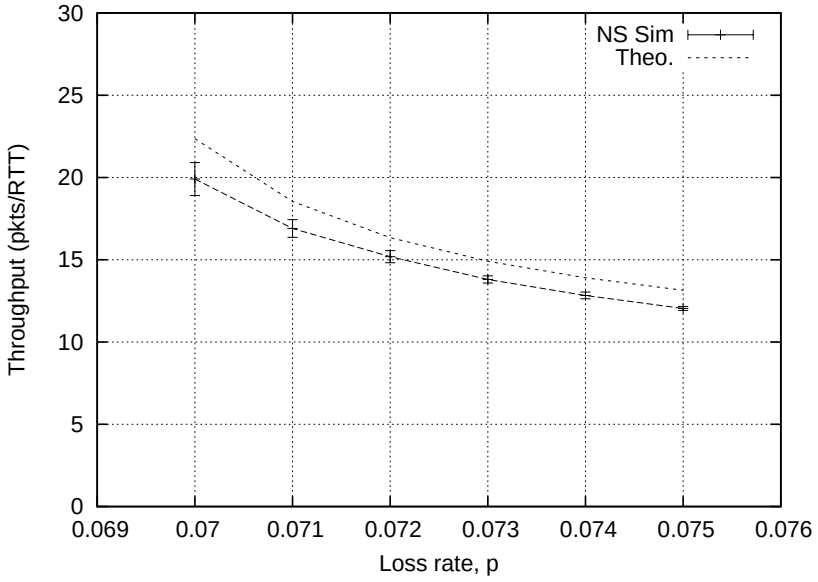


Figure 2.10: Throughput (pkts/RTT) versus loss rate, p, for Scenario (i).

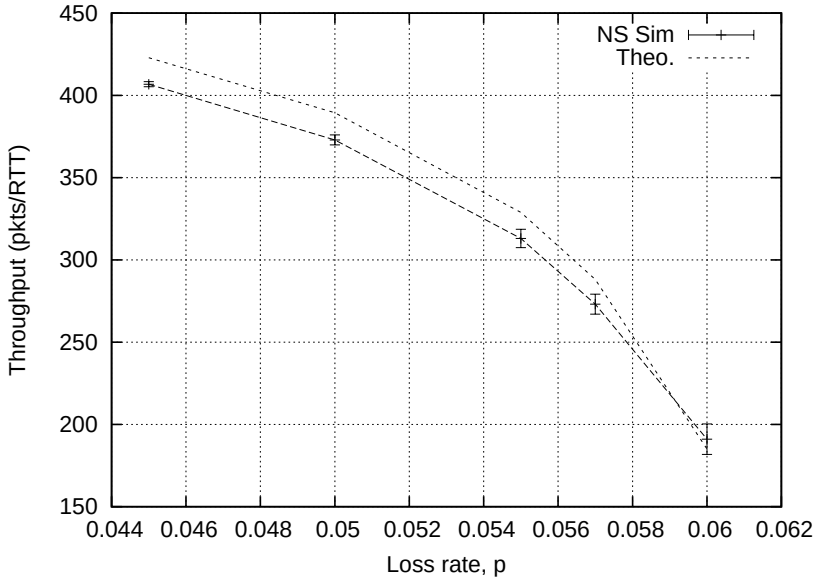


Figure 2.11: Throughput (pkts/RTT) versus loss rate, p, for Scenario (ii).

exact scenario (i.e., Scenario (iv)) fits well throughout the range of p. The throughput for Scenario (i) is shown for $p \geq 0.068$ because ζ (in Equation (2.45)) is > 1 for $p \geq 0.0673$.

The computation of E$[W]$ for different scenarios, requires different computational complexity. In order to obtain E$[W]$ of Scenario (i), we need to compute the largest root of the $k+1$ degree polynomial that is given by Equation (2.41). Once the largest root is obtained, we can compute all the moments of the window size using Equation (2.45). In Scenario (ii), the expected window size can be obtained using Equation (2.49) and Equation (2.51), and is relatively easier to compute than the E$[W]$ of Scenarios (i) and (iv). The expected window size of Scenario (iv) appears to require a large number of computations. Unlike in Scenario (i) and (iv), where the number of computations required is independent of the number of states of W, the computation of E$[W]$ in Scenario (iv) requires the computation of the probabilities that are given Equation (2.60).

2.10.3 Window-dependent loss process

We now compare the approximate throughput formula for the window dependent loss model that is given by Equation (2.76) with simulation results. The simulation setup is as before. In Figure 2.13 and Figure 2.14, the throughput is plotted as a function of the packet loss probability, q, for two different maximum window sizes, $W_{max} = 500$ and $W_{max} = 2000$. The approximation seems to give a good match for loss probabilities less than 0.001.

2.10.4 Window-independent first-order Markovian loss process

The effect of bursty losses on the behaviour of the AIMD algorithm was studied in [AAB00b]. It was observed that, in the absence of explicit window limitation, the throughput of TCP improved as the burstiness in the losses increased. In this subsection, we study the behaviour of the MIMD algorithm when burstiness increases in the losses.

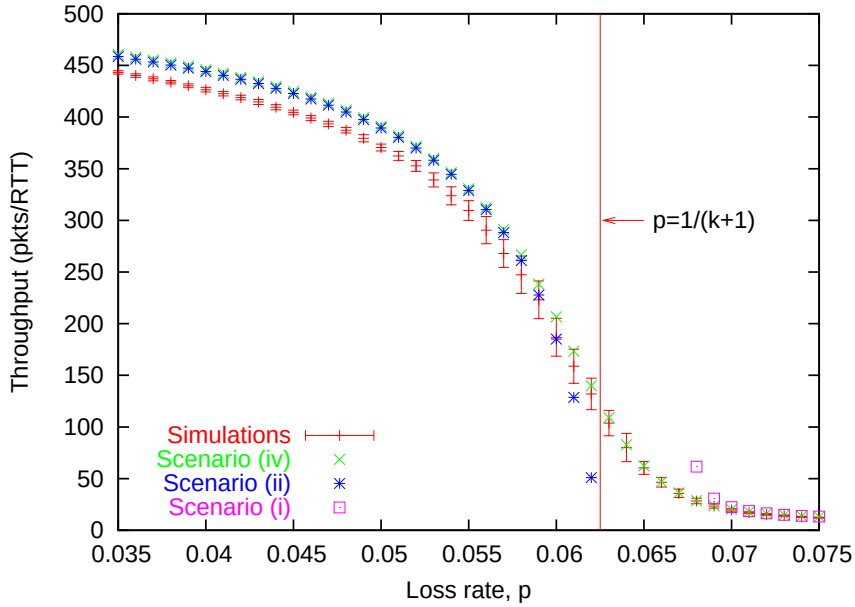


Figure 2.12: Throughput (pkts/RTT) versus loss rate, p, for Scenarios (i), (ii), and (iv)

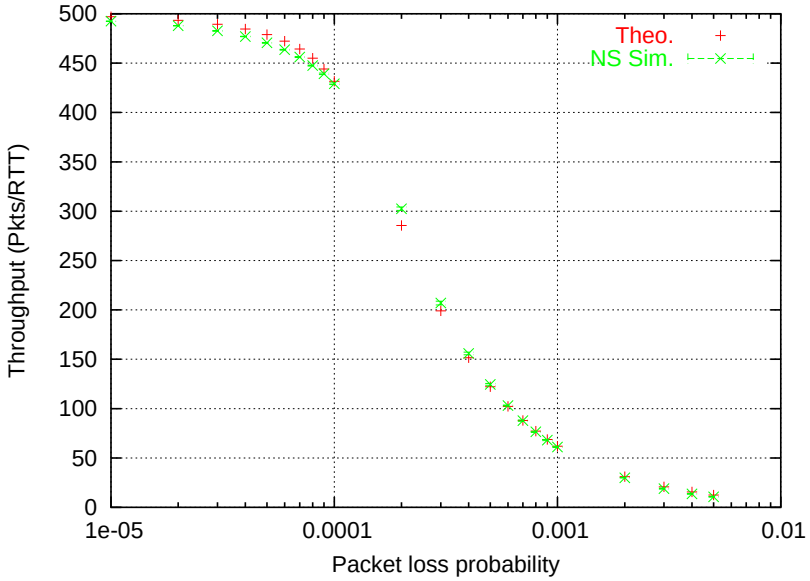


Figure 2.13: Throughput versus packet loss probability for the *window dependent loss* process. $W_{max} = 500$.

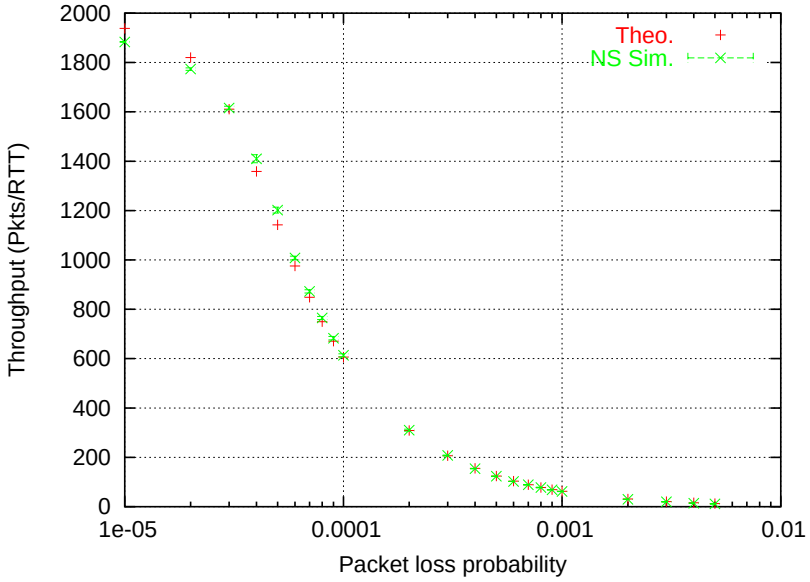


Figure 2.14: Throughput versus packet loss probability for the *window dependent loss* process. $W_{max} = 2000$.

In order to study the effect of burstiness, we fix the average loss rate, p_a, given in Equation (2.81), and vary the burstiness parameter b. As b increases, the probability that a loss would occur in the following RTT given that a loss occurred in the present RTT also increases. Therefore, losses tend to becomes more bursty as b increases. In Figure 2.15, the average loss rate is set to 0.02. We observe that the throughput decreases as the burstiness parameter increases. This observation may seem to be contrary to that in [AAB00b] where it was noted that bursty losses improve the throughput. However, we note that, unlike in [AAB00b], in this study the sender's window is limited by the receiver's advertised window, and this leads to the decrease in throughput.

As the average loss probability increases, the effect of bursty losses also increases. The graph of throughput versus the burstiness parameter for $p_a = 0.04$ is given in Figure 2.16.

2.11 Summary and conclusions

In this chapter we studied the performance of the MIMD algorithm in the presence of different loss processes. First, we presented a discrete time model for the evolution of the window size. The logarithm of the window size process was shown to be equivalent to the evolution of workload in a $G/G/1$ queue. We considered three loss processes:(i) *window-independent* loss process; (ii) *window-dependent* loss process; and (iii) *window-independent first-order Markovian* loss process. The moments of the window size process were obtained when the loss processes were *window-independent* and *window-independent first-order Markovian*. For the *window-dependent* loss process, we proposed an approximation using which we obtained the expected window size. The results obtained using the above analysis were validated through simulations.

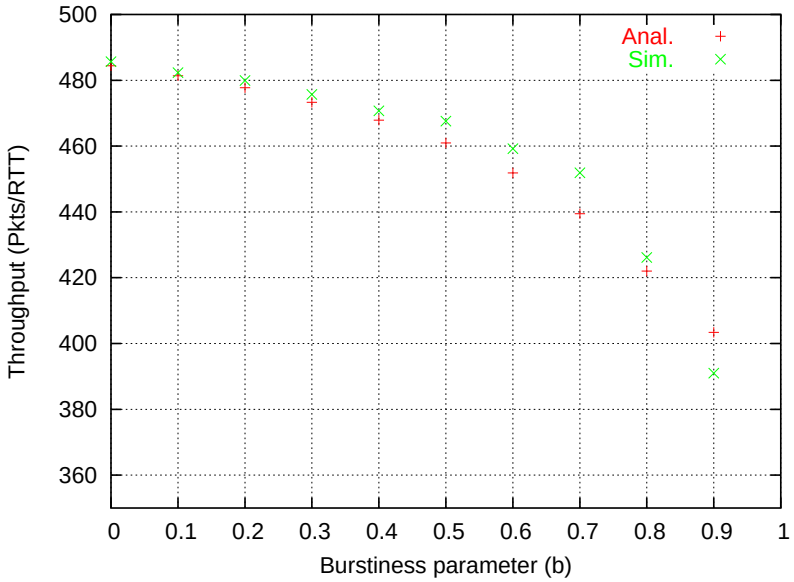


Figure 2.15: Throughput versus burstiness parameter. $p_a = 0.02$. $W_{max} = 500$.

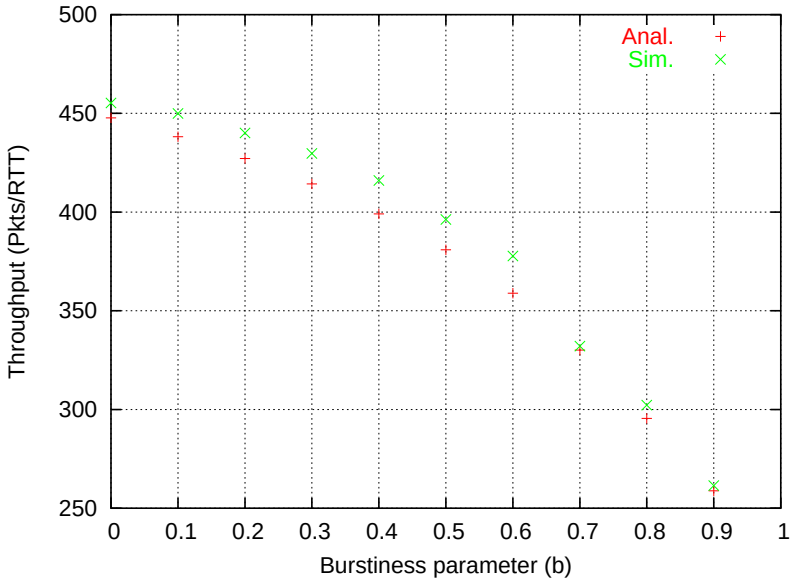


Figure 2.16: Throughput versus packet loss probability. $p_a = 0.04$. $W_{max} = 500$.

Chapter 3

Performance analysis of congestion control algorithms in continuous time

In the previous chapter we studied the behaviour of the window size of the MIMD algorithm in discrete time. In this chapter, we shall study window-based congestion control algorithms in continuous time. We shall start by considering a general increase algorithm and an instantaneous decrease algorithm in the presence of a window-dependent random loss process. We shall derive the Kolmogorov equation for the stationary distribution of the window size process. The window size process will be shown to be equivalent to the workload in a queue with workload-dependent arrival and service rates. This equivalence will enable us to give conditions under which two algorithms will have similar behaviour. We shall again derive the performance measures for the MIMD algorithm in continuous time.

The results in this chapter have appeared in [AAKP05].

3.1 Introduction

In the previous chapter, we studied the performance of the multiplicative increase multiplicative decrease (MIMD) congestion control algorithm in discrete time. We shall now study in continuous time the performance of a general increase and instantaneous decrease algorithm in the presence of a window-dependent loss process. The subject of this performance analysis was motivated by the following two observations. Firstly, while the behaviour of the additive increase algorithm in various loss environments has been a subject of research during the past few years (see, for example, [BHCKS04] for an extensive survey, and also Section 2.6), other increase algorithms have received less attention. With the rapid increase in network capacities, various increase algorithms have been proposed as alternatives to the additive increase algorithm. For example, CuBIC proposes a cubic increase algorithm and Scalable TCP proposes a multiplicative increase algorithm. A framework to analyse a general increase algorithm can help in comparing various proposals. Secondly, a congestion control has to operate in vastly heterogeneous environments. The heterogeneity can arise from, among others, geographical separation (which affects the round trip delay), nature of the underlying physical links (for example, a wireless link can suffer from sustained periods

of losses), and the buffer management scheme[1]. This heterogeneity can give rise to loss processes which are different from one another. Therefore, through this study we hope to provide a framework which can be used to analyse and compare different increase algorithms in the presence of different window-dependent loss rates. We would like to emphasize that we shall study only "loss-based" congestion control algorithms, which are described in Section 2.1.

3.2 System model

Let us consider a source which wants to send an infinite amount of data using a congestion control algorithm which operates in continuous time. As described in Section 2.1, a congestion control algorithm can be modelled using three building blocks. We shall now describe the increase algorithm, the decrease algorithm and the loss process, which together model the behaviour of the congestion control algorithm.

Let X_t denote the window size at time t. As has been assumed in related articles (see, for example, [KK02], [AABQ01a] and [BHCKS04]), we consider the window size as an infinitely divisible fluid. The window size is assumed to be bounded below by X_{min} and bounded above by X_{max}.

3.2.1 The increase algorithm

In the absence of losses in the interval $(\tau, \tau+\Delta)$, the dynamics of the window size can be described by the differential equation

$$\dot{X}_t = f(X_t), \quad \forall t \in (\tau, \tau+\Delta), \tag{3.1}$$

where $f(x) > 0, \forall x$, and f is a continuous function. We shall assume that, for a given value of X_τ, there exists a unique solution of the above differential equation in the interval $(\tau, \tau+\Delta)$. The window size in the continuous time system can be interpreted as an interpolation of the window size in the discrete time system. An additive increase algorithm that increases the window size by c units in every RTT can be interpolated by a linear increase algorithm with $f(X_t) = \frac{c}{RTT}$. Similarly, a multiplicative increase algorithm that increases the window size by a factor of α_d every RTT, can be interpolated by an exponential increase algorithm with $f(x) = \frac{\log[\alpha_d]}{RTT} x$.

3.2.2 The loss process

As mentioned in Section 2.1.2, we shall model the loss process observed by the source as a point process [BB03]. The points correspond to the arrival of loss notifications[2]. Specifically, we shall assume that the loss process is a simple Poisson point process with an intensity which depends only on X_t. Let N_t be the number of loss events observed by the source in the interval $(0, t]$. Using the Poissonian assumption, we have the following relations

$$\text{Prob}(N_{t+\delta} - N_t = 0) = 1 - \Lambda(X_t)\delta + o(\delta), \tag{3.2}$$

$$\text{Prob}(N_{t+\delta} - N_t = 1) = \Lambda(X_t)\delta + o(\delta), \tag{3.3}$$

[1] A buffer management scheme determines how a router drops a packet (either incoming or from the buffer) when the buffer starts to fill up.

[2] The loss could be due to either congestion or random events not related to congestion

where $\Lambda(x) > 0, \forall x$, is the window-dependent intensity. We shall assume that $\Lambda(X_t)$ is a continuous function. For example, the loss process $\Lambda(x) = \lambda x$ corresponds to a window-dependent loss process in which each packet is dropped with a probability λRTT [BHCKS04] (assuming λRTT < 1). The dependence of Λ on the window size could also be determined by some AQM scheme in the router which may decide to drop/mark packets according to some function of the window size.

3.2.3 The decrease algorithm

Let $\{\tau_n\}_{n\geq 1}$ denote the time instant when the nth loss notification arrives. On the reception of a loss notification, the algorithm decreases its window size instantaneously. Specifically,

$$X_{\tau_n +} = g(X_{\tau_n}), \tag{3.4}$$

where we assume that the function $g(x)$ satisfies the properties

1. $g(x) < x$, $x \in (X_{min}, X_{max}]$, $g(X_{min}) = X_{min}$, and
2. if $x_1 < x_2$, then either $g(x_1) < g(x_2)$ or $g(x_1) = g(x_2) = g(X_{min})$.

These properties imply that the set

$$s(x) = \{u \geq x : g(u) \leq x\}, \tag{3.5}$$

is connected. We shall use $g^{-1}(x)$ to denote $\sup s(x)$, i.e., $g^{-1}(x)$ is the maximal element from which X_t can jump to a window size of less than or equal to x. We also assume that any sample path of X_t is left continuous and has right limits.

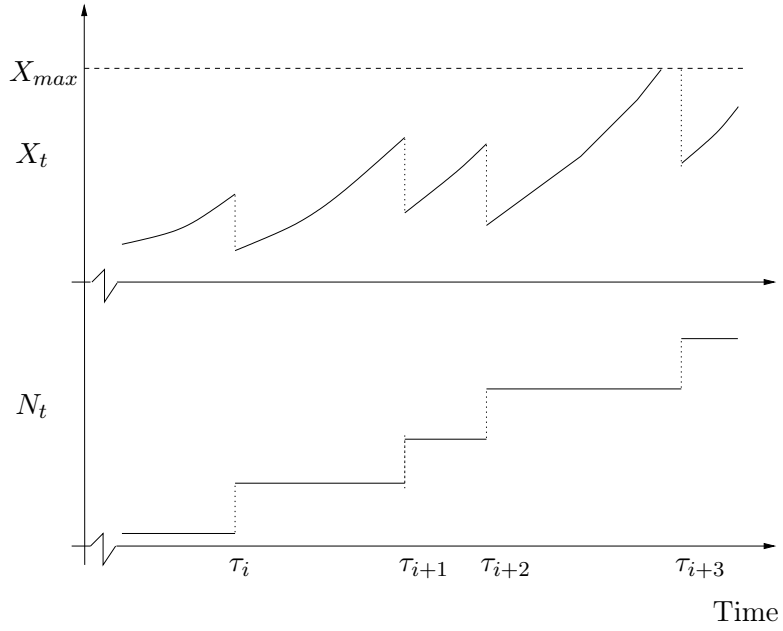


Figure 3.1: A Sample path of the process X_t (top figure) and N_t (bottom figure)

In Figure 3.1, we show a sample path of the processes X_t and N_t.

3.2.4 A few words on the modelling assumptions

As in Chapter 2, we shall make certain assumptions in the model. We shall consider a source which has an infinite amount of data to send. The congestion control algorithm is assumed to operate in the *congestion avoidance* phase. Therefore, the window behaviour is mainly determined by the increase algorithm, the decrease algorithm and the loss process as have been described above. We also make the following two assumptions:

1. *Constant round trip times*: The window dynamics in between loss instant depends on whether the round trip times are constant or not. As the round trip time increase the interpolation function also changes. For instance, in [KA04] it has been observed that a multiplicative increase algorithm changes to an additive increase algorithm as the buffer size increase.

2. *Instantaneous feedback:* The source is assumed to receive loss notification immediately and not after a delay of RTT. By means of this assumption, the loss process observed by a source depends on its current sending rate. If we were to assume delayed feedback then the loss process would depend on X_{t-RTT} making the analysis difficult.

3.2.5 Performance measures

As observed in Section 3.2, the behaviour of the window size is determined by the 3-tuple $< f,\ g,\ \Lambda >$, which we shall denote by "system". Let $F : X \mapsto \mathbb{R}_+$ be a function of the window size process. For a given system, we shall be interested in the time-average expected value of this function, $\mathrm{E}^{ta}[F(X)]$, i.e.,

$$\mathrm{E}^{ta}[F(X)] = \lim_{T\to\infty} \frac{1}{T} \int_0^T F(X_t)dt. \tag{3.6}$$

The function F could be suitably chosen so as to obtain the performance measure of interest. For example, F could be taken to be the identity function, $F(X) = X$, in which case one would obtain the time-average sending rate for the system. If F were to be the indicator function of the event $X_t \leq x$, $\mathbf{I}(X_t \leq x)$, then one would obtain the time-average distribution of the window size process, $\Pi(x)$, i.e.,

$$\Pi(x) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \mathbf{I}(X_t \leq x)dt. \tag{3.7}$$

Usually, the performance measures of a continuous time system are obtained by sampling the system at some discrete time instants. For example, a router in the Internet may monitor its queue size only at certain time intervals in order to reduce the overhead due to measurements. The performance measures obtained by observing the system at particular time instants are called as event-averages [MY95]. Let S_T denote the number of sampling events in the interval $(0, T]$. The process S_T is assumed to be such that $\lim_{T\to\infty} S_T \to \infty$. The event-average distribution function, $\widetilde{\Pi}(x)$, is obtained from the relation

$$\widetilde{\Pi}(x) = \lim_{T\to\infty} \frac{1}{S_T} \sum_{i=1}^{S_T} \mathbf{I}(X_i \leq x), \tag{3.8}$$

where X_i is value of X_t at the ith sampling instant.

3.3 Outline and contribution

In Section 3.4, we derive the Kolmogorov equation for the stationary distribution of the window size process of a given system. The solution of this equation will enable us to compute performance measures of interest. In Section 3.5, by means of the Kolomogorov equation, we shall provide conditions under which the window size process in two systems has the same behaviour. Furthermore, we shall show that the evolution of the window size is similar to the evolution of workload in queueuing system with workload-dependent arrival and service rates. Using this analogy, we shall give conditions under which two congestion control algorithms will have the same stationary distribution and, hence, have same performance.

In Section 3.6, we shall study a system with MIMD algorithm and constant loss intensity. We shall derive the performance measures by relating the window size with the workload in an equivalent queueing system. We shall also show that the window evolution of a general increase algorithm is equivalent to that of a linear increase algorithm. In Section 3.7 we shall derive the moments of the window size for the MIMD algorithm and linear loss intensity. We shall validate our analytical results by simulations in Section 3.8

We note that our analytical method is similar to that adopted in [AABQ01a] and [BHCKS04], where the authors only studied the window size of the AIMD algorithm.

3.4 The Kolmogorov equation

We first note that the process $\{X_t\}$ is a Markov process, i.e., the evolution of $\{X_t\}_{t>T}$ conditioned on $\{X_s\}_{s\leq T}$ is determined by X_T. We shall assume that the process $\{X_t\}$ is stationary and ergodic. Therefore, we have

$$\Pi(x) = \lim_{T\to\infty} \frac{1}{T}\int_0^T \mathbf{I}(X_t \leq x)dt = \lim_{t\to\infty} \text{Prob}(X_t \leq x), \tag{3.9}$$

We shall assume that $\Pi(x)$ is continuous everywhere except at X_{max}. Let $\pi(x)$ be the time-average density function of the stationary process, X. We note that, depending on $g(x)$, $\pi(x)$ may have points of discontinuity in the interval (X_{min}, X_{max}). In the rest of this chapter, we shall assume that the upper bound is due to the receiver's advertised window, $X_{max} = W_{recv}$. The relation between $\Pi(x)$ and $\Pi_{cl}(x)$, the limiting distribution when X_{max} is due to the congestion losses, is given in [AABQ01a], and we state it below.

$$\begin{aligned} \Pi_{cl}(x) &= \frac{\Pi(x)}{1-\phi(X_{max})}, \\ \text{where} \quad \phi(X_{max}) &= 1 - \int_{X_{min}}^{X_{max}-} d\Pi(x), \end{aligned} \tag{3.10}$$

is the atom of the stationary distribution at the state X_{max}.

Let $\mathcal{M}$ denote the set $[g(X_{max}), X_{max})$, and let $\mathbf{I}(\mathcal{M})$ be the indicator function of $x \in \mathcal{M}$. We shall assume that the functions $\Pi(\cdot)$ and $\widetilde{\Pi}(\cdot)$ satisfy the condition

$$\Pi(X_{max}) = 1, \text{ and } \widetilde{\Pi}(X_{max}) = 1. \tag{3.11}$$

Proposition 3.4.1 *For a system $<f, g, \Lambda>$, the unique time-average stationary distribution function, $\Pi(x)$, is a solution of the equation*

$$f(x)\frac{d\Pi(x)}{dx} = \Lambda(X_{max})\phi(X_{max})\mathbf{I}(\mathcal{M}) + \int_x^{g^{-1}(x)} \Lambda(u)d\Pi(u), \tag{3.12}$$

$$\textit{where } \phi(X_{max}) = 1 - \int_{X_{min}}^{X_{max}-} d\Pi(x). \tag{3.13}$$

Proof. Please see Appendix B.1. ∎

Similarly, we can also obtain the event-average stationary distribution of the window size process embedded just before loss instants.

Proposition 3.4.2 *For a system $<f, g, \Lambda>$, the unique event-average stationary distribution function, $\widetilde{\Pi}(x)$, is a solution of the equation*

$$\widetilde{\Pi}(x) = \mathsf{A}(x - g(X_{max}))\tilde{\phi}(X_{max})\mathbf{I}(m) + \int_{X_{min}}^{g^{-1}(x)} \mathsf{A}(x - g(u))d\widetilde{\Pi}(u). \tag{3.14}$$

$$\textit{where } \tilde{\phi}(X_{max}) = 1 - \int_{X_{min}}^{X_{max}-} d\widetilde{\Pi}(x), \tag{3.15}$$

$$\textit{and } \mathsf{A}(x - g(\cdot)) = 1 - \exp\left(-\int_{g(\cdot)}^{x} \frac{\Lambda(y)}{f(y)}dy\right). \tag{3.16}$$

Proof. Please see Appendix B.2. ∎

For a function $F(X)$, we define

$$\mathrm{E}[F(X)] = \int_{X_{min}}^{X_{max}-} F(x)d\Pi_i(x) + F(X_{max})\phi(X_{max}). \tag{3.17}$$

3.5 Relation between two systems

Let us consider two systems, $<f_1, g_1, \Lambda_1>$ and $<f_2, g_2, \Lambda_2>$ which have no lower bound and no upper bound on the window size, i.e., $X_{min} = 0$, $X_{max} = \infty$. We shall assume that the window size process in each system has a unique stationary density, $\pi_i(\cdot)$, $i = 1, 2$. We shall provide a condition under which π_1 is related to π_2. Since $X_{max} = \infty$, the point mass at X_{max}, $\phi(X_{max})$, is equal to zero. From Equation (3.12), the stationary density of the window size in System i, π_i, satisfies the equation

$$f_i(x)\pi_i(x) = \int_x^{g_i^{-1}(x)} \Lambda_i(u)\pi_i(u)du, \quad x \in (0, \infty), \quad i = 1, 2. \tag{3.18}$$

Proposition 3.5.1 *If $<f_1, g_1, \Lambda_1>$ and $<f_2, g_2, \Lambda_2>$ are two systems such that*

1. $g_1(x) = g_2(x) = g(x)$, $\forall x$, *and*
2. $\frac{f_1(x)}{\Lambda_1(x)} = \frac{f_2(x)}{\Lambda_2(x)} = \kappa(x)$, $\forall x$,

then

$$\frac{\pi_1(x)}{\pi_2(x)} = C\frac{\Lambda_2(x)}{\Lambda_1(x)} = C\frac{f_2(x)}{f_1(x)}, \quad \forall x, \tag{3.19}$$

where $C = \frac{E[\Lambda_1(X)]}{E[\Lambda_2(X)]}$.

Proof. We note that we have assumed $f(x), \Lambda(x) > 0, \ \forall x$. From Equation (3.18) and Condition 1, $\pi_i(x)$ satisfies

$$\frac{f_i(x)}{\Lambda_i(x)} \frac{\Lambda_i(x)\pi_i(x)}{\mathrm{E}[\Lambda_i(X)]} = \int_x^{g^{-1}(x)} \frac{\Lambda_i(z)\pi_i(z)}{\mathrm{E}[\Lambda_i(X)]} dz, \quad x \in (0, \infty), \quad i = 1, 2. \tag{3.20}$$

Let us define $\hat{\pi}_i(x) = \frac{\Lambda_i(x)\pi_i(x)}{\mathrm{E}[\Lambda_i(X)]}$. We note that $\int_x \hat{\pi}(x)dx = 1$. Using Condition 2,

$$\kappa(x)\hat{\pi}_i(x) = \int_x^{g^{-1}(x)} \hat{\pi}_i(z)dz, \quad i = 1, 2. \tag{3.21}$$

Since $\hat{\pi}_1$ and $\hat{\pi}_2$ are solutions to the same equation which has a unique solution, we can conclude that $\hat{\pi}_1(x) = \hat{\pi}_2(x), \ \forall x$, thereby obtaining the relation in Equation (3.19). ■

We can also relate the density function of the stationary window size process in System i observed just before loss instants, i.e., the event-average density function, $\tilde{\pi}_i(\cdot)$.

Proposition 3.5.2 *Under conditions stated in Proposition 3.5.1, the event-average distribution of the stationary window size process in the two systems is related as*

$$\widetilde{\Pi}_1(x) = \widetilde{\Pi}_2(x), \quad \forall x. \tag{3.22}$$

Proof. For System i, the point mass at $X_{max} = \infty$, $\tilde{\phi}_i(X_{max})$ will be zero. Therefore, from Equation (3.14), $\widetilde{\Pi}_i(x)$ is a solution of the equation

$$\widetilde{\Pi}_i(x) = \int_{X_{min}}^{g^{-1}(x)} \mathsf{A}_i(x - g(u)) d\widetilde{\Pi}_i(u). \tag{3.23}$$

Since $\mathsf{A}_i(x - g(u))$ depends on the ratio $\frac{\Lambda(x)}{f(x)}$, which is same for both the systems, $\mathsf{A}_1(x - g(u)) = \mathsf{A}_2(x - g(u))$. Therefore, $\widetilde{\Pi}_1(x)$ and $\widetilde{\Pi}_2(x)$ are solutions to the same equation, which has a unique solution. Hence, we obtain the relation in Equation (3.22). ■

In order to illustrate the results in this section, we shall make use of two particular systems:

1. System A will refer to a system with an AIMD algorithm and a constant loss intensity, i.e., $<\alpha, \beta x, \lambda>$; and

2. System M will refer to a system with an MIMD algorithm and a linear loss intensity, i.e., $<\alpha x, \beta x, \lambda x>$.

The window size process in both the systems is assumed to have the same lower and upper bounds. The stationary density of the window size processes in these two systems will be denoted by $\pi_A(\cdot)$ and $\pi_M(\cdot)$, respectively. The System A corresponds to the AIMD algorithm, which is used in standard TCP, and a constant loss intensity, whereas the System M corresponds to the MIMD algorithm, which has been proposed in Scalable TCP, and a linear loss intensity.

The result in Proposition 3.5.1 gives conditions under which we can analyse one system using the analysis of a related system. For example, System A and System M satisfy the conditions in Proposition 3.5.1. If we assume that the window size process in one of the systems has a unique stationary distribution then we can obtain the time-average stationary distribution of the window size process in the other system. Similarly, using Proposition 3.5.2, we can relate the event-average stationary distribution of the window size process in the two systems.

Next, we study the the general increase and multiplicative decrease algorithm with a constant decrease factor of β. Here we can get some more equivalence results for two related systems when $X_{min} = 0$ and $X_{max} < \infty$.

3.5.1 A queueing model for multiplicative decrease algorithms

Let us consider a system $<f, g, \Lambda>$ in which $g(x) = \beta x$, where $\beta < 1$. We note that in this section $X_{min} = 0$ and $X_{max} < \infty$. We now introduce the transformation

$$Z_t = \log[X_{max}] - \log[X_t]. \tag{3.24}$$

In the transformed system, Z_t takes values in $[0, \infty)$. Here we are assuming that $X_{min} = 0$. In order to find the stationary distribution, $\Pi_Z(\cdot)$, when $X_{min} > 0$, we can first find the stationary distribution with $X_{min} = 0$ and then use the truncation method [Asm03, Chapter 14]. We note that this method can be used only when the unbounded process has a unique stationary distribution.

Let $<f_Z, g_Z, \Lambda_Z>$ denote the system for the process $\{Z_t\}$. From Equation (3.1) and Equation (3.24), we have the relations,

$$f_Z(z) = -\frac{f(X_{max}\exp(-z))}{X_{max}\exp(-z)}, \quad g_Z(z) = z + \log[1/\beta], \text{ and } \quad \Lambda_Z(z) = \Lambda(X_{max}\exp(-z)). \tag{3.25}$$

A sample path of the process $\{X_t\}$ and the transformed process $\{Z_t\}$ is shown in Figure 3.2. The evolution of the process $\{Z_t\}$ resembles the evolution of workload in a queue with workload-dependent arrival rate, $\Lambda_Z(\cdot)$, workload-dependent service rate, $|f_Z(\cdot)|$, and service requirements of constant size, $\log[1/\beta]$. Such a queueing system has been studied in [BBBK04]. The condition[3] for the existence of a unique stationary distribution of the workload in the queueing system (and, hence, for the process Z_t) is given in [BBBK04], and can be stated as

$$\limsup_{z\to\infty} \log[1/\beta]\frac{\Lambda_Z(z)}{|f_Z(z)|} < 1. \tag{3.26}$$

The stability condition for System A requires

$$\limsup_{z\to\infty} \log[1/\beta]\frac{\lambda X_{max}}{\alpha}\exp(-z) < 1, \tag{3.27}$$

[3]The condition is necessary only for unbounded systems, i.e., systems with $X_{min} = 0$. A system with $X_{min} > 0$ is always stable because the process cannot escape from the closed and bounded set $[X_{min}, X_{max}]$.

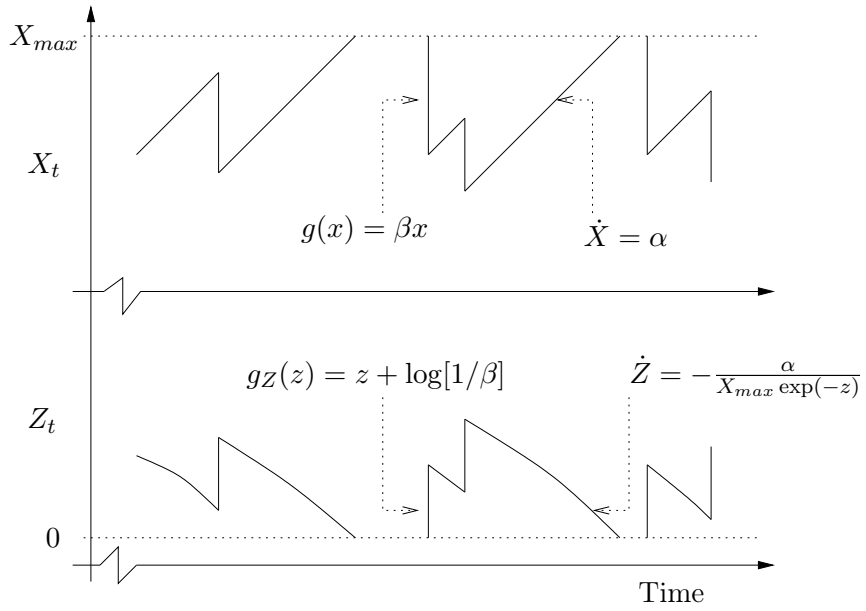


Figure 3.2: The process X_t which has an instantaneous multiplicative decrease (top figure) is transformed into a process which an instantaneous constant increase (bottom figure).

which is indeed satisfied for fixed values of the parameters α, β, and λ. Using the relation in Equation (3.19), we can say that System M is also stable for fixed values of α, β, and λ. For a stable system, the authors of [BBBK04] also provide a condition under which the stationary distribution of the workload in the queueing system has an atom at state 0. The condition for the existence of an atom at state 0 requires

$$\int_0^z \frac{\Lambda_Z(u)}{|f_Z(u)|} du \ < \ \infty, \quad \forall \ z \in (0, \infty). \tag{3.28}$$

The existence of an atom at state 0 in the queueing system is equivalent to the existence of an atom at state X_{max} in the window size system. For System A (and consequently for System M) this condition translates into

$$\frac{\lambda X_{max}}{\alpha} \int_0^z \exp(-z) \ < \ \infty, \quad \forall \ z \in (0, \infty), \tag{3.29}$$

which is satisfied for fixed values of the parameters. Hence, the stationary distributions of the window size processes in Systems A and M will have an atom at state X_{max}, i.e., $\phi_A(X_{max}) > 0$ and $\phi_M(X_{max}) > 0$.

For the multiplicative decrease algorithm with $X_{max} < \infty$, Proposition 3.5.1 can be restated as

Proposition 3.5.3 *If* $<f_1, g_1, \Lambda_1>$ *and* $<f_2, g_2, \Lambda_2>$ *are two systems such that*

1. $g_1(x) = g_2(x) = \beta x, \ \forall x,$ *and*

2. $\frac{f_1(x)}{\Lambda_1(x)} = \frac{f_2(x)}{\Lambda_2(x)}, \forall x,$

then

$$\frac{\pi_1(x)}{\pi_2(x)} = C\frac{f_2(x)}{f_1(x)} = C\frac{\Lambda_2(x)}{\Lambda_1(x)}, \tag{3.30}$$

$$\textit{where} \quad C = \frac{\Lambda_1(X_{max})\phi_1(X_{max})}{\Lambda_2(X_{max})\phi_2(X_{max})} \ \textit{if} \ \phi_i(X_{max}) > 0. \tag{3.31}$$

Proof. Using the transformation in Equation (3.24), each system has an equivalent queueing system in which the service requirements are the same in both the systems. The proof then follows from [BBBK04, Theorem 3.1]. ■

The equivalence between the window size in a multiplicative decrease algorithm and the workload in queue with workload-dependent arrival and service rates also helps us to derive the relation between the stationary distribution of the window size observed just before loss instant in both the systems, i.e., the event-average stationary distribution. Let $\tilde{\pi}(\cdot)$ denote the event-average density of the window size process in System i.

Proposition 3.5.4 *Under conditions stated in Proposition 3.5.3,*

$$\tilde{\pi}_1(x) = \tilde{\pi}_2(x), \quad \forall\, x \in (0, X_{max}), \tag{3.32}$$

$$\tilde{\phi}_1(X_{max}) = \tilde{\phi}_2(X_{max}). \tag{3.33}$$

Proof. As in the proof of Proposition 3.5.3, the proof follows from the equivalence relation of the window size process and the workload process in a queue with workload-dependent arrival and service rates, and from [BBBK04, Theorem 3.3]. ■

If one of the systems has a constant loss intensity, then by the PASTA property [Wol89] the time-average stationary distribution of the window size in that system is equal to the stationary distribution observed just before loss instants in either system. For the two system example, Proposition 3.5.4 implies that System A and System M have the same stationary distribution of the window size process observed just before loss instants. Since the loss intensity is constant in System A, this distribution is also the (time-average) stationary distribution in System A. Since System A has been analysed in [AABQ01a], we can use PASTA and Equation (3.32) to obtain the stationary distribution of the window size observed just before loss instants in System M. We shall illustrate these results with simulative examples in Section 3.8.

Remark 3.5.1 *The equivalence between a window size system and a queueing system is, in general, not restricted to window size systems with multiplicative decrease algorithms. For a general instantaneous decrease function, $g(x)$, if we can solve the functional equation*

$$h(g(x)) = h(x) + 1, \tag{3.34}$$

for function h along with the conditions that h is a decreasing function with continuous first derivative[4] *and $h : (0, X_{max}] \mapsto [0, \infty)$, then we can use the transformation $z = h(x)$ and obtain an equivalent queueing system with workload-dependent arrival rate $\Lambda(h^{-1}(z))$, workload-dependent service rate $f(h^{-1}(z))\frac{dh(x)}{dx}|_{x=h^{-1}(z)}$, and constant service requirements of unity. The Equation (3.34) is referred to as Abel's functional equation. For the multiplicative decrease algorithm, the function $h(x) = \frac{\log[X_{max}]-\log[x]}{\log[1/\beta]}$ solves Equation (3.34).*

[4] Since $z = h(x)$, $\frac{dz}{dt} = f(x)\frac{dh(x)}{dx}$. Since $f(x)\frac{dh(x)}{dx}$ is the service rate in the queueing system it should be negative.

3.6 System with MIMD algorithm and a constant loss intensity

In this section we shall analyse the behaviour of the window size process in a system with MIMD algorithm and a constant loss intensity. The system is represented as $<\alpha x, \beta x, \lambda>$. This system is the continuous time equivalent of the system studied in Section 2.7

3.6.1 No upper bound on window size

First, we assume $X_{min} > 0$ and $X_{max} = \infty$. We introduce the transformation

$$Y_t = \frac{\log[X_t] - \log[X_{min}]}{\alpha}. \tag{3.35}$$

The domain of Y is the set $[0, \infty)$. The transformed window size system can be described by

$$f_Y(y) = 1, \quad g_Y(y) = y - \frac{\log[1/\beta]}{\alpha} \text{ and } \Lambda_Y = \lambda. \tag{3.36}$$

The transformed process is a linear increase constant decrease process. The dynamics of $\{Y_n\}$ are as follows. In between loss instants the process $\{Y_t\}$ increases with rate unity. The inter-arrival times for the loss notifications are exponentially distributed with intensity λ. On the arrival of a loss notification the transformed window size is decreased by an amount $\theta = \log[1/\beta]/\alpha$ subject to a lower bound of zero. In order to obtain the stationary distribution of $\{Y_t\}$, we embed the process just after loss instants. Let $\{Y_n\}$ denote the embedded process. The recursive equation for the evolution of $\{Y_n\}$ as

$$Y_{n+1} = \max(Y_n + A_n - D_{n+1}, 0), \tag{3.37}$$

where A_n denotes the interval of time between the nth and the $(n+1)$th loss instants, and D_n is the amount of decrease at the $(n+1)$th loss instant. The random variables $\{A_n, n \geq 1\}$ are independent and exponentially distributed with intensity λ. The random variables $\{D_n, n \geq 1\}$ are identical and equal to θ. Thus, the evolution of $\{Y_n\}$ is similar to the evolution of the workload observed just before loss instants in a $D/M/1$ queue. The stationary distribution of the transformed window size process observed just after loss instants is given by [Kle75]

$$\text{Prob}(Y > y) = \left(1 - \frac{s^*}{\lambda}\right) e^{-s^* y}, \tag{3.38}$$

where $-s^*$ is the root of the equation $s + \lambda = \lambda e^{s\theta}$ in $\text{Re}(s) < 0$. The stability condition for the workload process of this $D/M/1$ queue and, equivalently, of the window size process $\{Y_t\}$ is

$$\frac{1}{\lambda} < \theta. \tag{3.39}$$

From Equation (3.35) and Equation (3.38), the complementary distribution function the stationary window size process, X, observed just after loss instants is given by

$$\tilde{\Pi^c}_X(x) = \left(1 - \frac{s^*}{\lambda}\right) \left(\frac{x}{X_{min}}\right)^{-\frac{s^*}{\alpha}}, \quad x > X_{min}. \tag{3.40}$$

There is an atom of size $\frac{s^*}{\lambda}$ at X_{min}.

Next, we obtain the the (time-average) stationary distribution of the transformed window size process. We first note that the transformed window size just before the $(n+1)$th loss instant, Y_{n+1}^-, is given by

$$Y_{n+1}^- = Y_n + A_n. \tag{3.41}$$

Since A_n is exponentially distributed with parameter λ,

$$\begin{aligned}
\text{Prob}(Y_{n+1}^- > y) &= \lambda \int_0^\infty \text{Prob}(Y_n > y-u | A_n = u) e^{-\lambda u} du && (3.42)\\
&= \lambda \int_0^\infty \text{Prob}(Y_n > y-u) e^{-\lambda u} du && (3.43)\\
&= \lambda \int_y^\infty e^{-\lambda u} du + \lambda \int_0^y \text{Prob}(Y_n > y-u) e^{-\lambda u} du && (3.44)\\
&= e^{-\lambda y} + \lambda \left(1 - \frac{s^*}{\lambda}\right) e^{-s^* y} \int_0^y e^{-(\lambda - s^*)u} du && (3.45)\\
&= e^{-s^* y}. && (3.46)
\end{aligned}$$

Using PASTA property, the (time-average) stationary distribution of the transformed window size process is the same as distribution observed just before loss instants. From Equation (3.35) and Equation (3.46), the (time-average) complementary distribution function of the stationary window size process, X, can be obtained as

$$\Pi_X^c(x) = \left(\frac{x}{X_{min}}\right)^{-\frac{s^*}{\alpha}}, \quad x \geq X_{min}. \tag{3.47}$$

3.6.2 No lower bound on window size

We now assume $X_{min} = 0$ and $0 < X_{max} < \infty$. As in for the system with no upper bound, we transform the window size using the transformation

$$Y_t = \frac{\log[X_{max}] - \log[X_t]}{\alpha}. \tag{3.48}$$

The domain of Y is the set $[0, \infty)$. The evolution of Y can be described by

$$f_Y(y) = 1, \quad g_Y(y) = y + \frac{\log[1/\beta]}{\alpha}, \quad \text{and } \Lambda_Y(y) = \lambda. \tag{3.49}$$

We note that, unlike $\{X_t\}$, the process $\{Y_t\}$ is decreasing subject to a lower bound in between loss instants. The inter-arrival times between losses are exponentially distributed with intensity λ, and upon the reception of a loss notification the process $\{Y_t\}$ jumps up by a constant $\theta = \log[1/\beta]/\alpha$. The evolution of the process $\{Y_t\}$ is similar to the evolution of the workload in a $M/D/1$ queue with arrival rate λ and service time θ. The condition for stability of the $M/D/1$ queue and, equivalently, of the transformed system is

$$\theta < \frac{1}{\lambda}. \tag{3.50}$$

The Laplace transform of the stationary distribution of the workload in the $M/D/1$ described above and, equivalently, of the transformed process is given by [Kle75]

$$\mathbf{Y}(s) = (1 - \lambda\theta)\frac{s}{s - \lambda + \lambda e^{-s\theta}}. \tag{3.51}$$

From Equation (3.48) and Equation (3.51), we can obtain the moments of the stationary window size process as

$$\mathrm{E}[X^j] = X_{max} Y(\alpha^{-j}). \tag{3.52}$$

3.6.3 A transformation for the window size process

In the previous two subsections we noted that by transforming the process $\{X_t\}$, we obtained a process with linear dynamics in between loss instants. This method of transforming X_t to a process with linear dynamics in between loss instants can be extended to general increase algorithms. Let us consider a system $<f, g, \Lambda>$ with window size process X_t. We assume $0 < X_{min} < X_{max} < \infty$. Let $\mathsf{f}(x)$ be a function such that

$$\mathsf{f}(x) = \int \frac{1}{f(x)} dx. \tag{3.53}$$

We now define the process

$$Y_t = \mathsf{f}(X_t). \tag{3.54}$$

The transformed system can be described by

$$f_Y(y) = 1, \quad g_Y(y) = g(\mathsf{f}^{-1}(y)), \quad \text{and} \quad \Lambda_Y(y) = \Lambda(\mathsf{f}^{-1}(y)), \tag{3.55}$$

with $0 < \mathsf{f}(X_{min}) \le Y_t \le \mathsf{f}(X_{max}) < \infty$. The process $\{X_t\}$ is now transformed to a system with linear increase, general decrease, and a general loss intensity. We can thus study the performance of a general increase algorithm by studying the performance of a linear increase algorithm albeit with a different decrease algorithm and loss intensity. In Section 3.6.1 and Section 3.6.2, we made the transformation $\mathsf{f}(x) = \log[x]$ to obtain a linear increase algorithm.

3.7 System with MIMD algorithm and a linear loss intensity

In this section we shall obtain moments of the window size process in a system with MIMD algorithm and a linear loss intensity ($<\alpha x, \beta x, \lambda x>$), i.e., the moments of the window size process in System M. We shall assume that $X_{min} = 0$ and $X_{max} = \infty$. We can expect this system to approximate the scenario when the lower and upper bounds are rarely attained. From Equation (3.12), the Kolmogorov equation for this system is

$$\alpha x \pi_M(x) = \int_x^{x/\beta} \lambda u \pi_M(u) du. \tag{3.56}$$

We now multiply both sides of the equation by x^{j-1} and integrate over $x \in (0, \infty)$ to obtain

$$\alpha \int_0^\infty x^j \pi_M(x) = \lambda \int_0^\infty x^{j-1} \int_x^{x/\beta} u \pi_M(u) du \, dx, \quad j = 0, 1, 2, \ldots \tag{3.57}$$

which simplifies to

$$\alpha E[X^j] = \lambda \int_0^\infty u \int_{\beta u}^u x^{j-1} dx \, \pi_M(u) du \quad j = 0, 1, 2, \ldots \tag{3.58}$$

$$= \begin{cases} \lambda \log[1/\beta] E[X], & j = 0; \\ \lambda \frac{(1-\beta^j)}{j} E[X^{j+1}], & j = 1, 2, \ldots. \end{cases} \tag{3.59}$$

Therefore,

$$E[X] = \frac{\alpha}{\lambda \log[1/\beta]}, \tag{3.60}$$

$$E[X^{j+1}] = \frac{\alpha^j j!}{\lambda^j \prod_{i=1}^{j} (1 - \beta^i)} E[X] \quad j = 1, 2, \ldots. \tag{3.61}$$

Thus, all the moments are finite even when $X_{min} = 0$ and $X_{max} = \infty$. We note that the relation $E[X] \propto 1/\lambda$ was also obtained using an approximation in Section 2.8.

3.8 Simulation results

In this section we compare the analytical results obtained in this chapter with simulation results. The simulations were performed using the open source simulator *ns-2* [MF]. The simulation setup has a source and destination node connected using a link of capacity 150Mbps. The round trip time is assumed to be 120ms. The packet size for data was taken to be 1040bytes. The BDP of this network is roughly 2250 packets. We set the receiver's advertised window to 500, i.e., $X_{max} = 500$. With this setting the RTT is almost a constant. In the simulation, we used the SACK option for both standard TCP as well as Scalable TCP.

3.8.1 Relation between two systems

First, we compare the equivalence results obtained in Section 3.5. We continue with the two-system example of System A and System M.

System A is the continuous time equivalent of the standard TCP with SACK, which uses the AIMD algorithm. In the absence of losses, standard TCP increases the window size by one packet every RTT. In continuous time, this increase corresponds to to setting $f_A(x) = 1/RTT$. The multiplicative decrease parameter for standard TCP is 0.5. The decrease parameter remains unchanged for both the discrete time as well as the continuous time systems, i.e., $g_A(x) = 0.5x$. The loss process for this system has a constant loss intensity of $\Lambda_A(x) = \lambda_A/RTT$.

System M is the continuous time equivalent of the Scalable TCP algorithm with SACK, which uses the multiplicative increase multiplicative decrease algorithm. In the absence of losses, Scalable TCP increases the window size by a factor of 1.01 every RTT. In continuous time, this corresponds to setting $f_M(x) = \log[1.01]x/RTT$. The multiplicative decrease parameter for Scalable TCP was set to 0.5. The author in [Kel03b] proposes a decrease parameter of 0.875. Since we need the two systems to have the same decrease algorithm, we set the parameter to 0.5, i.e., $g_M(x) = 0.5x$. The loss process in this system will be $\Lambda_M(x) = \lambda_M x/RTT$.

Since we are only interested in the ratio $f(x)/\Lambda(x)$, in the following discussion we will scale $f(x)$ and $\Lambda(x)$ by RTT. Thus, when we say that $f_A(x) = 1$ the actual function used in the

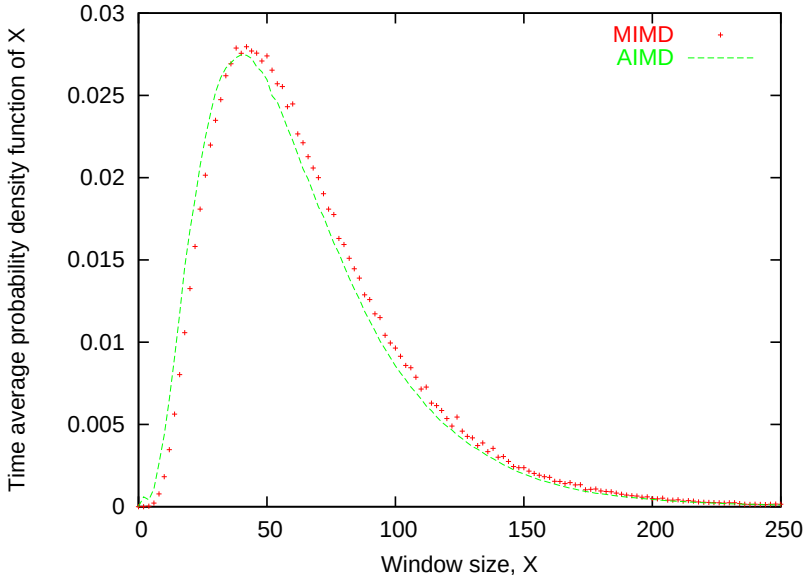


Figure 3.3: Time-average density function of the window size, X.

simulations will be $f_A(x) = 1/RTT$. We scale the functions so as to simplify the notation. With this description, the systems can be summarized as

$$\text{System A} =< 1, 0.5, \lambda_A > \text{ and } \text{System M} =< 0.01x, 0.5, \lambda_M x >, \tag{3.62}$$

where we have made the approximation $\log[1.01] \sim 0.01$.

First, we validate the analytical results obtained in Proposition 3.5.1 and Proposition 3.5.2. We assume $\lambda_A = 0.03$ and $\lambda_M = 0.0003$. The two systems now satisfy the conditions in Proposition 3.5.1. Therefore, we need to verify

$$\pi_A(x) = \frac{\mathrm{E}[\Lambda_A(X)])}{\mathrm{E}[\Lambda_M(X)]} \frac{\Lambda_M(x)}{\Lambda_A(x)} \pi_M(x) \tag{3.63}$$

$$= \frac{x\pi_M(x)}{\mathrm{E}[X_M]}, \tag{3.64}$$

where $\mathrm{E}[X_M]$ is the expected window size in System M. In Figure 3.3 we plot $\pi_A(x)$ and $\frac{\pi_M(x)x}{47.67}$, where $\pi_A(\cdot)$ and $\pi_M(\cdot)$ were obtained through simulations. In this particular example, through simulations it was observed that $\mathrm{E}[X_M] = 47.67$. We see that there is a good match between the analytical prediction and the simulation results. Now, we compare the event-average distribution function in the two systems. In Figure 3.4 we plot the functions $\widetilde{\Pi}^c_A(x)$ and $\widetilde{\Pi}^c_M(x)$. As was predicted in Proposition 3.5.2 the two functions follow each other very closely.

We note that in the above example the loss intensity was such that the lower bound, X_{min}, and the upper bound, X_{max}, were rarely reached. Hence, we could apply the results obtained for the unbounded system.

Next, we validate the results for the two systems when there is a positive point mass at state X_{max}, i.e., $\phi_A(X_{max}), \phi_M(X_{max}) > 0$. The corresponding analytical results were obtained in Proposition 3.5.3 and Proposition 3.5.4. We assume $\lambda_A = 0.008$ and $\lambda_M = 0.00008$. Again, these

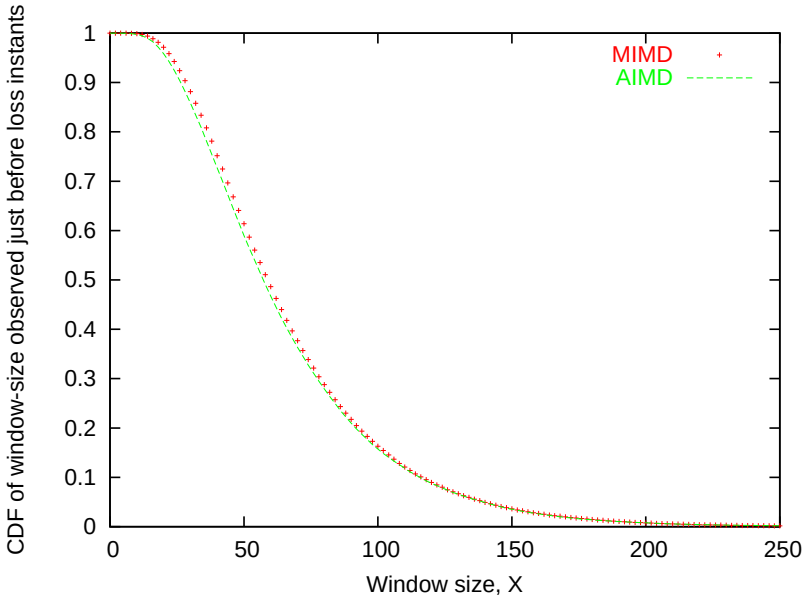


Figure 3.4: Event-average complementary distribution function of the window size, X.

two systems verify the conditions in Proposition 3.5.1. Therefore, we need to verify

$$\pi_A(x) = \frac{\Lambda_A(X_{max})\phi_A(X_{max})}{\Lambda_M(X_{max})\phi_M(X_{max})}\frac{\Lambda_M(x)}{\Lambda_A(x)}\pi_M(x) \tag{3.65}$$

$$= \frac{\phi_A(X_{max})}{X_{max}\phi_M(X_{max})}x\pi_M(x) \tag{3.66}$$

$$\tag{3.67}$$

From the simulations, we observed that $\phi_A(X_{max}) = 0.052$ and $\phi_M(X_{max}) = 0.0184$. In Figure 3.5, we plot $\pi_A(\cdot)$ and the RHS of Equation (3.66) in which all the parameters were obtained through simulations. We note that the density function has discontinuities at X_{max} and at $g(X_{max})$ (i.e., at $0.5X_{max}$). A good match is observed between the analytical prediction and the simulation results. In Figure 3.6, we compare the event-average complementary distribution function for the two systems obtained through simulations. The simulation results validate the analytical result obtained in Proposition 3.5.4.

3.8.2 System with MIMD algorithm and a constant loss intensity

Next, we validate the results obtained in Section 3.6. Here we only consider the performance measures for System M. The parameters for this system were set to $<0.01x, 0.8.0.47>$. We note that this set of parameters satisfies the stability condition in Equation (3.39). In Figure 3.7, we plot the time-average complementary distribution function of the stationary window size process. The analytical solution is obtained from Equation (3.47). The complementary distribution function of the stationary window size process observed just after loss instants in shown in Figure 3.8. The corresponding analytical solution was obtained in Equation (3.40). A good match is observed between analysis and simulations.

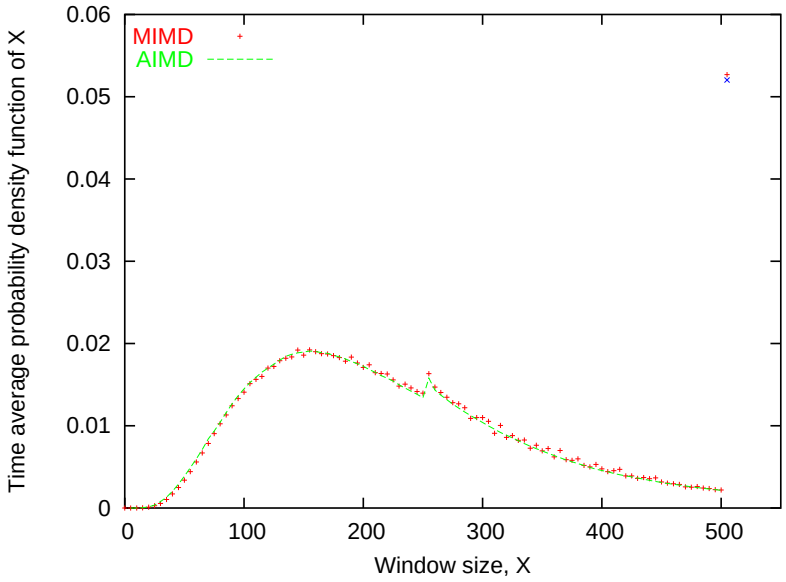


Figure 3.5: Time-average density function of the window size, X.

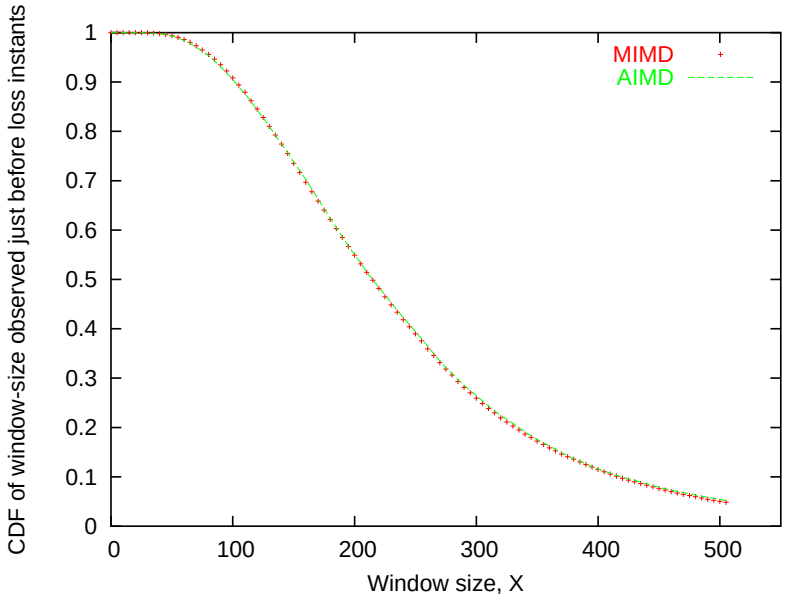


Figure 3.6: Event-average complementary distribution function of the window size, X.

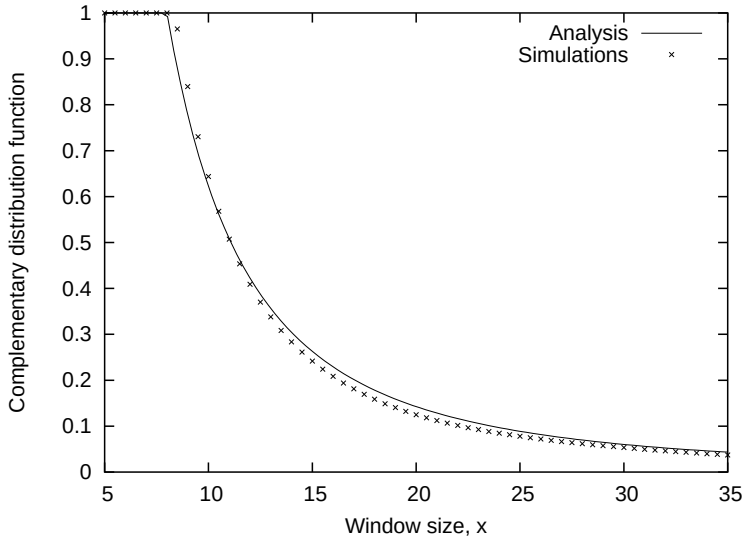


Figure 3.7: Time-average complementary distribution function of the window size, X. $\frac{s^*}{\alpha} = 2.12$.

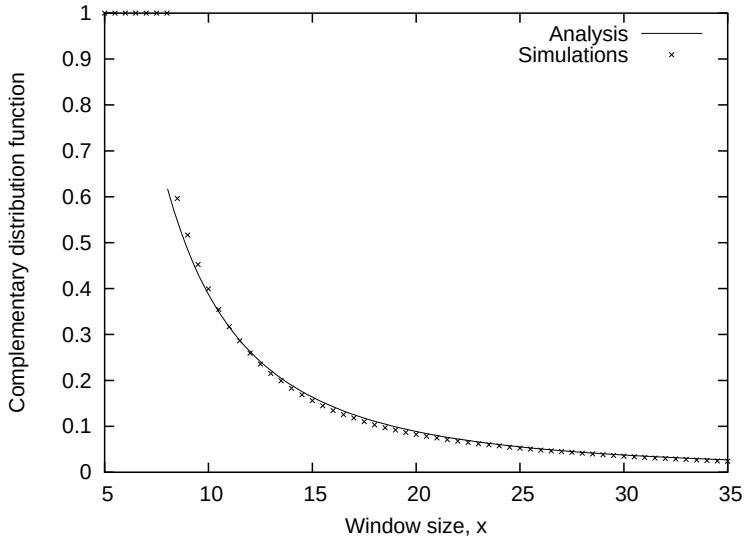


Figure 3.8: Event-average complementary distribution function of the window size, X. $\frac{s^*}{\alpha} = 2.12$.

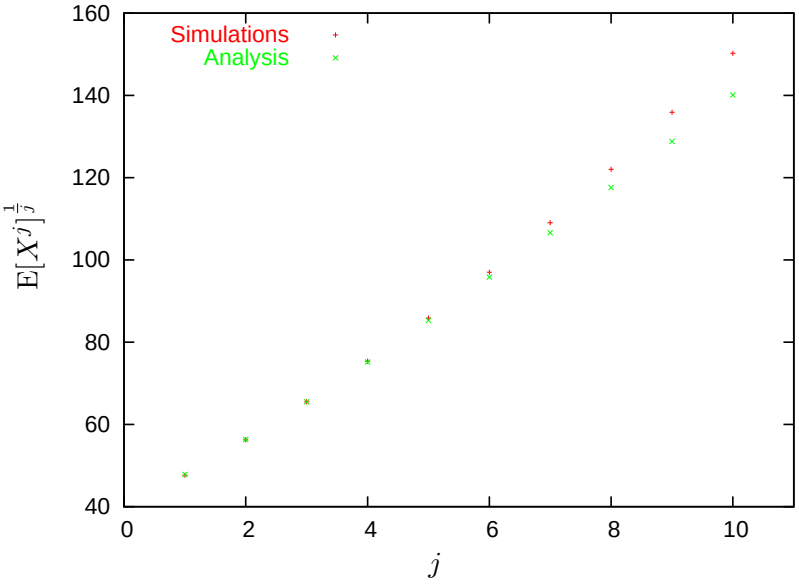


Figure 3.9: Moments of the window size of the MIMD algorithm with a linear loss intensity.

3.8.3 System with MIMD algorithm and a linear loss intensity

Finally, we validate the expression for the moments of the window size process of the system considered in Section 3.7. We note that in this system there is no lower bound and no upper bound on the window size. For the simulations, we set the parameter values to $<0.01x, 0.5, 0.0003x>$. In Figure 3.9, we plot the function $\mathrm{E}[X_M^j]^{1/j}$ as obtained through simulations and from Equation (3.60) and Equation (3.61).

3.9 Summary and conclusions

In this chapter we studied a continuous time model for the window size of a congestion control algorithm in the presence of a window-dependent Poisson loss process. We provided conditions under which two congestion control algorithms have the same window size behaviour, thereby enabling us to analyze one system using the analysis of the other. It was observed that the event-average distribution of the window size in a system with AIMD algorithm and a constant loss intensity is the same as the event-average distribution of the window size in a system with MIMD algorithm and a linear loss intensity. We also obtained the performance measures for the MIMD algorithm with window-independent and window-dependent loss processes. The average sending rate of the MIMD algorithm in the presence of a linear loss intensity, i.e, λx, was observed to be inversely proportional to λ.

Chapter 4

Some fairness properties of the MIMD congestion control algorithm

In the previous two chapters we studied the behaviour of the window size from the perspective of an individual source in the network. On most occasions a source will find itself sharing the network resources with other sources. The sharing of common resource by different individuals gives rise to questions about fairness of the resource allocation. In this chapter, we shall first study how sources with MIMD congestion control algorithm share the capacity of a link. Then, we shall study how sources using the MIMD algorithm and the AIMD algorithm share the capacity of a common link.

The results in this chapter have appeared in [AAP05].

4.1 Introduction

Let us consider two sources which want to transfer data across a common link of capacity C as shown in the Figure 4.1. The share of the bandwidth obtained by either source will depend on the congestion control algorithm that it employs and the parameters[1] of the algorithm that it sets. We shall say that the sources are homogeneous if both of them employ the same congestion control algorithm with identical parameters. Otherwise, we shall say that the sources are heterogeneous. The Internet admits a diverse set of sources which are quite often heterogeneous. In an environment in which individuals share a resource, each individual may be interested in knowing the proportion of the resource that it obtains. Since a source using the Internet relies on a congestion control algorithm to determine its share, one of the design goals of a congestion control algorithm is to ensure that the source obtains a "fair"[2] share of the network resources. For homogeneous sources sharing a single link, under certain conditions it was shown that the AIMD algorithm converges to an allocation of equal share [CJ89]. This was one of the reasons why the AIMD algorithm was chosen for implementation in standard TCP. For homogeneous sources in a multiple-link network, the AIMD algorithm has been observed to be proportionally fair [KMT98], i.e., the source obtains a share which is in proportion to the number of links that it uses. For sources with heterogeneous RTTs, [VBB00] showed that the AIMD algorithm converged to a variant of proportional fairness

[1] The parameters here refer to the parameters of the increase and decrease algorithms, and the round trip times.

[2] We shall use the fairness index of [CJ89] as a measure of fairness.

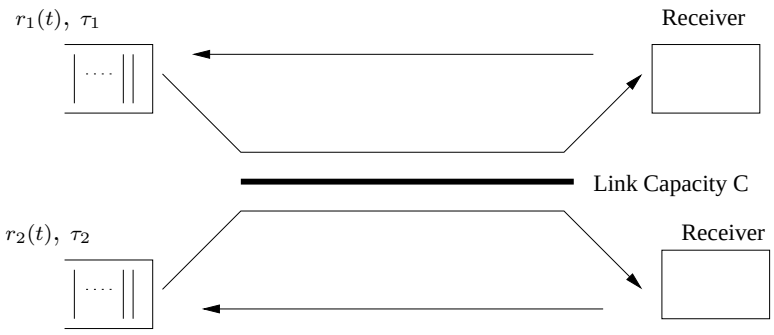


Figure 4.1: The network model for multiple users

with a bias towards flows with smaller RTTs. Although the AIMD algorithm has certain fairness properties, it was observed to be inefficient in utilizing the capacity in networks with large capacities. It may be agreed that any proposal to modify the current algorithm would need to have fairness properties which are similar, if not better. In this chapter we follow up the analysis of the average sending rate in the previous two chapters with analysis of the fairness properties of the MIMD algorithm on a *single link*. First, we study the *intra-algorithm* fairness properties of the MIMD algorithm. We shall distinguish two scenarios in which to study the *intra-algorithm* fairness properties. In the first scenario, we shall study the fairness properties for homogeneous sources, i.e., for sources that have identical parameters for the algorithm and identical RTTs. In the second scenario we shall study the fairness properties for heterogeneous sources, i.e., for sources that have the identical parameters for the algorithm but different RTTs. Next, we shall study the *inter-algorithm* fairness[3] properties of the MIMD algorithm. Any new proposal that will be implemented is likely to be incrementally deployed. Therefore, there may exist situations when sources using the MIMD algorithm and the AIMD algorithm will share a link. We shall study how the MIMD algorithm shares the bandwidth with the AIMD algorithm.

4.2 The system model

In this section we shall describe the system model (the loss process, in particular). Since there are two sources instead of one, as was the case in the previous chapters, the loss process observed by the sources could be correlated. This correlation can affect the fairness properties of an algorithm as we shall see later.

We use the following notation in this chapter. The sets $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{Z}$ will denote the set of real numbers, the set of positive real numbers, and the set of integers, respectively. A function from $\mathbb{R}$ to $\mathbb{R}$ will be denoted using sans serif font. For example, $\mathsf{x}(t)$ denotes a function defined for all real values of t. A function from $\mathbb{Z}$ to $\mathbb{R}$ will be denoted using italic fonts as $x()$. Usually, $x(n)$ would be the value of $\mathsf{x}(t)$ at the nth sampling instant. A vector $r = (r_1, r_2, \ldots, r_N)$ will denote an N dimensional row vector. Its transpose will be denoted by r'. Also, we shall use the term "session" to mean an instance of a given algorithm. The term source will be used for someone who makes use of one or more instances (i.e., sessions) of the same algorithm.

Our system model is based on the model originally considered in [CJ89]. Let us consider two sessions which share a link of capacity C as shown in Figure 4.1. Let $\mathsf{x}(t) \equiv (\mathsf{x}_1(t), \mathsf{x}_2(t))$ be the

[3]We shall also refer to this as bandwidth sharing between AIMD and MIMD algorithms.

rate[4] vector at time t, where $\mathsf{x}_1(t)$ and $\mathsf{x}_2(t)$ denote the instantaneous rates of session 1 and of session 2, respectively. The set of feasible rate vectors, $\{(\mathsf{x}_1, \mathsf{x}_2) | \mathsf{x}_1 + \mathsf{x}_2 \leq C; \mathsf{x}_1, \mathsf{x}_2 \geq 0\}$ is shown as the shaded area in Figure 4.2. The line $\mathsf{x}_1(t) + \mathsf{x}_2(t) = C$ is called the efficiency line. On this

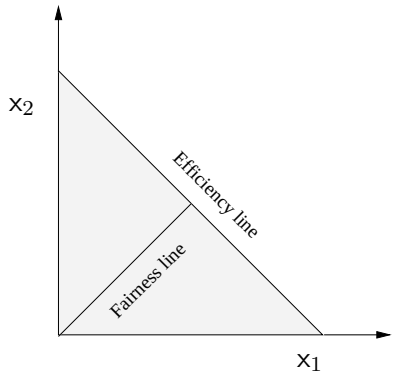


Figure 4.2: The rate allocation vector

line the available capacity is fully utilized. The line $\mathsf{x}_1(t) = \mathsf{x}_2(t)$ is called the fairness line. On this line both the sessions obtain the same rates, and hence the bandwidth sharing is said to be fair.

We shall assume that the increase algorithm is updated in continuous time as was the case in Chapter 3. Specifically, for the MIMD algorithm, the rate increases as

$$\dot{\mathsf{x}}_i(t) = \alpha_i \mathsf{x}_i(t), \quad i = 1, 2, \tag{4.1}$$

$$\text{where } \alpha_i = \frac{\log[\alpha_d]}{RTT_i}, \tag{4.2}$$

and $\alpha_d > 1$ is the increase parameter of the discrete-time algorithm. For example, for Scalable TCP $\alpha_d = 1.01$. The decrease algorithm is instantaneous and multiplicative, i.e.,

$$\mathsf{x}_i(\text{just after a loss instant}) = \beta \cdot \mathsf{x}_i(\text{at that loss instant}), \quad i = 1, 2, \tag{4.3}$$

where we have assumed that any sample path of $\mathsf{x}_i(t)$ is left-continuous with right-hand limits.

4.2.1 The loss process

As was done in previous chapters, we shall assume that the congestion control algorithm relies on binary feedback (i.e., presence or absence of congestion) from the network to adapt the sending rate, and to reduce congestion. The presence of congestion in the network is signalled by either dropping or marking packets. A congestion signal can occur not only when the link capacity is achieved but could also be initiated before that. For example, a congestion signal is sent as a consequence of a packet drop when the link buffer overflows (i.e., when the capacity is achieved), or when a router using AQM schemes such as RED or ECN decides to respectively drop or mark a packet (i.e., a congestion signal is initiated before the capacity is achieved). In reaction to the congestion signal, the algorithm reduces its sending rate. We shall use the term "loss"[5] to refer to any signal that causes a reduction in the sending rate.

[4] We note that here we are using the term "rate" instead of the "window size" as was done in previous chapters. The window size and the rate are related as: rate $= \frac{\text{window size} \times \text{packet size}}{\text{RTT}}$. We use the the term "rate" because the fairness index is defined in terms of the rates observed by each session.

[5] We shall use the terms losses and congestion signals interchangeably.

A congestion signal is said to be *synchronous* when all the sessions sharing the link receive that congestion signal at the same time instant. A congestion signal is said to be *asynchronous* when some, but not all, sessions receive that congestion signal at the same time instant. We shall assume that a *synchronous* congestion signal is generated when the capacity is achieved. In addition to these congestion signals, we shall assume that the network generates congestion signals as a Poisson process of intensity λ before the capacity is achieved. Each congestion signal that is generated by the network is *synchronous* with probability $1-\epsilon$ and is asynchronous with probability ϵ. We shall use two models to determine which of the sessions receives the asynchronous congestion signal:

1. The *rate-dependent loss model* in which the probability that a session receives the congestion signal is proportional to its current rate at the congestion instant [VBB00], [BHCKS04], i.e.,

$$\text{probability session } i \text{ receives asynchronous congestion signal} = \epsilon \frac{\mathsf{x}_i(t)}{\mathsf{x}_1(t)+\mathsf{x}_2(t)},\ i=1,2. \tag{4.4}$$

2. The *rate-independent loss model* in which the probability that a session receives a congestion signal is independent of the session's rate [KMT98],[BH02], i.e.,

$$\text{probability session } i \text{ receives asynchronous congestion signal} = \frac{\epsilon}{2},\quad i=1,2. \tag{4.5}$$

Remark 4.2.1 *When we say "a congestion signal is generated" we shall refer to any congestion signal. The term "a congestion signal is generated by the network" will exclusively refer to congestion signals which are generated before the capacity is achieved.*

In Figure 4.3, the evolution for the sending rate is shown as a function of time when two sources share a link.

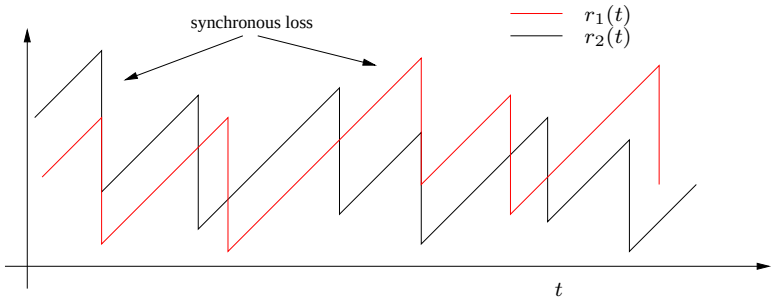


Figure 4.3: Window evolution for two sources sharing a link.

Remark 4.2.2 *The above loss model could be extended to N sessions. For example, in [BH02] the authors propose a method to compute the probability that k out of N sessions receive the congestion signal.*

4.2.2 A few words on the modelling assumptions

As in Chapter 3, we shall make the following assumptions:

1. *Constant round trip times*; and

2. *Instantaneous feedback:* This assumption is required only when we consider the rate-dependent loss model.

These two assumptions were also made in Section 3.2.4 and are made in this model for the similar reasons.

4.2.3 Fairness index

Let N be the number of sessions. The fairness index is a function $F : \mathbb{R}_+^N \mapsto \mathbb{R}_+$ using which one can compare two feasible rate vectors. For a given feasible rate vector r, in [CJ89] the authors considered the index

$$F = \frac{(\sum_{i=1}^{N} r_i)^2}{N \sum_{i=0}^{N} r_i^2}. \tag{4.6}$$

A property of this index is that $F \leq 1$ with equality if and only if $r_1 = r_2 = \ldots = r_N$[6]. Also, $F \geq 1/N$ with equality if, for some i, $r_i \neq 0$ and $r_j = 0, \forall j \neq i$. For a rate vector r, the preceding observations (also in [CJ89]) can be interpreted as: (i) if $F = 1$ then we can say that the rate vector is a "fair" allocation; whereas (ii) if $F = 1/N$ then we can say that only one session gets the resource, thus the rate vector can be termed as "unfair" to other sessions.

In our model, the rate vector, $\{\mathsf{x}(t), t \geq 0\}$ is a stochastic process, which may or may not converge to a stationary and ergodic stochastic process. In order to take into account the dynamical nature of the rate vector, we reformulate the fairness index as

$$F_* = \inf_{\mathsf{x}(0)} \lim_{T \to \infty} \frac{1}{T} \int_0^T \frac{(\sum_{i=1}^{N} \mathsf{x}_i(t))^2}{N \sum_{i=0}^{N} \mathsf{x}_i(t)^2} dt. \tag{4.7}$$

Since the process $\{\mathsf{x}(t)\}$ need not be ergodic, we shall take the infimum over $\mathsf{x}(0)$ to consider the *worst-case* fairness index is possible for the process $\{\mathsf{x}(t)\}$ that could obtained. The subscript $*$ will denote the system under consideration. In general, the fairness index is associated with a particular system which comprises of sessions and the loss process. The sessions could be either homogeneous or heterogeneous. Therefore, we shall associate the fairness index with a system and not just with an algorithm, for example, sessions using MIMD algorithm and a synchronous loss process can be considered as an example of a system. We shall consider three systems: (i) a system with sessions using MIMD algorithm and a synchronous loss process, i.e., $* = sy$; (ii) a system with sessions using MIMD algorithm and a rate-independent loss process, i.e., $* = ri$; and (iii) a system with MIMD algorithm and a rate-dependent loss process, i.e., $* = rd$. If, in system $*$, the process $\{\mathsf{x}(t), t \geq 0\}$ converges to a stationary and ergodic process $\tilde{\mathsf{x}}$ then F_* can be computed as

$$F_* = \frac{1}{N} \mathrm{E}\left[\frac{(\sum_{i=1}^{N} \tilde{\mathsf{x}}_i)^2}{\sum_{i=1}^{N} \tilde{\mathsf{x}}_i^2} \right]. \tag{4.8}$$

In the rest of this chapter we shall refer to F_*, which is defined in Equation (4.7), as the *fairness index*.

[6]This property can be obtained through the Cauchy-Schwartz inequality, $|r \cdot \mathbf{1}| \leq \|r\| \cdot \|\mathbf{1}\|$, where $\mathbf{1} = (1, 1, \ldots, 1)$.

4.3 Related literature and contributions

In [CJ89], the authors studied a discrete-time model version of the above model with synchronous control signals[7]. The sessions reacted to control signals by adapting their rates in the following way:

$$x(n+1) = \begin{cases} b_I x(n) + a_I & \text{for an increase (or a no congestion) signal,} \\ b_D x(n) + a_D & \text{for a decrease (or a congestion) signal,} \end{cases}$$

where a_I, b_I, a_D and b_D are constants and the sampling is done just after the reception of control signals. In TCP NewReno, for example, arrival of ACKs can be considered as increase signals, and arrival of duplicated ACKs or a time-out can be considered to be decrease signals. In [CJ89] it has been argued that for convergence to the fairness line, the increase algorithm has to be multiplicative and additive (i.e., $a_I > 0$ and $b_I \geq 1$) and the decrease algorithm has to be multiplicative ($b_D < 1$ and $a_D = 0$). However, this result was derived under the assumption of synchronous control signals. That is, both the sessions receive control signals at the same instant and both of them either increase, or decrease their rates simultaneously. Indeed, under these two assumptions, the rate vector for the MIMD algorithm stays on a line joining the origin to the initial rate vector, and hence does not converge to fairness. Therefore, the fairness index for this system[8], i.e., F_{sy}, is $1/N$, which is the minimum possible value. In [BB01], the authors extend the model in [CJ89] to include binomial increase and decrease functions, i.e., they consider the following rate adaptation:

$$x(n+1) = \begin{cases} b_I x(n) + a_I w^{-k} & \text{for an increase (or a no congestion) signal,} \\ b_D x(n) + a_D w^{l} & \text{for a decrease (or a congestion) signal,} \end{cases}$$

We note that $k = -1$ and $l = 1$ corresponds to the MIMD algorithm. They conclude that for the rate vector to converge to the fairness line, the conditions $k+l > 0$, $k \geq 0$ and $l \geq 0$ must be satisfied. As in [CJ89], the authors of [BB01] also assume that the congestion signals are synchronous. In [Gor04], the authors argue that under a more realistic assumption of rate-dependent congestion signals, the MIMD algorithm also converges to the fairness line. In the first part of the chapter, we first show that, for sessions with the same RTT, $F_{rd} > F_{sy}$, i.e., in a system with sessions using MIMD algorithm, the fairness index can be improved by introducing rate-dependent losses. We obtain the expressions for the long term fairness index and the mean time to reach the fairness line. We note that our model is stochastic, hence any performance measures will be an expectation of a function of a random variable.

In [XHR04] it was argued that for sessions with different RTTs, Scalable TCP (or, the MIMD algorithm) is extremely unfair, i.e., the session with the smallest RTT obtained the whole capacity, thereby giving a fairness index of $1/N$. We show that, even in systems with heterogeneous MIMD sessions, the fairness index can be improved by introducing rate-dependent losses with intensity *greater* than a certain number. We then show, through simulations, that the results obtained for two sessions also hold for N sessions.

In the second part of this chapter, we shall consider several sessions sharing a common link on which losses are due to buffer overflow and are, can be considered to be synchronous. In [HMM00] such a scenario was considered for sessions using the AIMD algorithm only. The authors of [BB00] studied algorithms other than AIMD in a synchronous scenario. However, they considered scenarios in which sources used the same algorithm. In [XHR04], fairness issues were

[7] A control signal can be either a signal to increase the rate or a signal decrease the rate.

[8] $N = 2$ in this case.

studied for different TCP versions using high-speed networks. The RTT-fairness, i.e., bandwidth sharing among sessions with different RTTs, was compared for different proposals of TCP. Here, too, the fairness was studied among sessions using the same algorithm, i.e., the above mentioned studies considered only *intra-algorithm* fairness. The *inter-algorithm* bandwidth sharing has been studied in [GM04] for sessions of TCP NewReno [APS99], TCP Westwood+ [GM02] and TCP Vegas [BP95], mainly through simulations. In [MLAW99], the authors study the coexistence of sources using TCP Vegas and TCP Reno. Bandwidth sharing of standard TCP sessions and non-TCP sessions has been studied in the context of compatibility, or "friendliness" of such flows with standard TCP. In order to study *inter-algorithm* fairness properties of the MIMD algorithm, we shall consider a heterogeneous scenario where different sessions may use either the AIMD or the MIMD algorithms. This type of scenarios may be of interest in the future when, for example, sessions using Scalable TCP and standard TCP may share the same link. We shall obtain expressions for the equilibrium throughput (i.e., the average number of bits sent per unit of time) and the window size of the sessions and compare the analytical results with simulations.

We also mention that fairness issues for congestion control algorithms on multiple links have been studied in, among others, [KMT98], [MW00],[VBB00]. For example, in [MW00], the authors consider a set of sources-destination pairs which share a path (i.e., a set of links with other source-destination pairs. For a given feasible [9] rate vector, x, the authors define a global utility which is a function of x, a vector p and a scalar α. They say that a rate vector is (p, α) proportionally fair[10] if it maximizes the global utility subject to the rate vector being feasible. We note that the authors in [KMT98], [MW00] first define a fairness criterion and then design an algorithm which converges to the fair allocation vector. Our study concerns source-destination pairs which share a single link as shown in Figure 4.1. In single link scenarios, the proportionally fair vector is the one which allocates equal rates to all the sources. In this regard, our model is restrictive. However, we consider a slightly different problem: we are given a congestion-control algorithm that has been shown to be quite unfair under certain conditions (see [CJ89], [XHR04], and the discussion in the preceding paragraphs). Our aim is to provide conditions under which the fairness index of this algorithm can be improved. Also, we analyse the model in a stochastic setting as opposed to the deterministic differential equation approximation approach used in [KMT98] and [MW00]. The contributions of this work include: (i) the conditions under which the fairness index of the MIMD algorithm can be improved; and (ii) the results of bandwidth sharing between AIMD and MIMD sources.

4.4 Outline

The rest of the chapter is organized as follows. First, we shall study the *intra-algorithm* properties of the MIMD algorithm. In Section 4.5, we consider a homogeneous scenario wherein two MIMD sessions with the same set of parameters share a link. We show that the fairness index of this system can be improved by introducing rate-dependent losses at a non zero intensity. In Section 4.6 we shall study the heterogeneous scenario wherein the two sessions have may different parameters. We shall show that the fairness index of such a system can only be improved by introducing rate-dependent losses at an intensity greater than a threshold intensity. Next, in Section 4.7 and Section 4.8 we study the bandwidth sharing between AIMD and MIMD sessions when losses are synchronous. We

[9] The feasibility conditions include non-negativity of the rate vector, and that the sum of the rates of source-destination pairs sharing a link should not exceed the link capacity.

[10] In [KMT98], the authors studied the $(p, 1)$ proportional fairness.

Table 4.1: Reaction to congestion signals

congestion signal	$\mathsf{x}_1(t_n+)$	$\mathsf{x}_2(t_n+)$	
$(0,1)$	$\mathsf{x}_1(t_n)$	$\beta \cdot \mathsf{x}_2(t_n)$	} asynchronous congestion signal
$(1,0)$	$\beta \cdot \mathsf{x}_1(t_n)$	$\mathsf{x}_2(t_n)$	
$(1,1)$	$\beta \cdot \mathsf{x}_1(t_n)$	$\beta \cdot \mathsf{x}_2(t_n)$	synchronous congestion signal

show that the AIMD sessions obtain a share which is independent of the link capacity, and that the rest of the capacity is utilized by the MIMD sessions. Finally, we present the conclusions in Section 4.9.

4.5 Intra-algorithm fairness of the MIMD algorithm: homogeneous sessions

We consider two sessions using the MIMD algorithm which share a link of capacity C. At time t, the rates of the two sessions are denoted by $\mathsf{x}(t) \equiv (\mathsf{x}_1(t), \mathsf{x}_2(t))$. The feasible set of rate vectors for this system is

$$\{(\mathsf{x}_1(t), \mathsf{x}_2(t)) : \mathsf{x}_1(t) + \mathsf{x}_2(t) \leq C, \;\; \mathsf{x}_1(0), \mathsf{x}_2(0) > 0\}, \;\; \forall t \geq 0. \tag{4.9}$$

Remark 4.5.1 *The initial rate vector* $\mathsf{x}(0)$ *is taken such that* $\mathsf{x}_i(0) > 0$. *For the MIMD algorithm if the initial rate is zero then the vector stays at zero which is not desirable.*

We shall assume that, in the absence of congestion signals, the source increases its sending rate exponentially, i.e.,

$$\mathsf{x}_i(t+\tau) = \exp(\alpha_i \tau) \cdot \mathsf{x}_i(t), \quad i = 1, 2, \tag{4.10}$$

$$\text{where } \alpha_i = \frac{\log[\alpha_d]}{\tau_i}, \tag{4.11}$$

τ_i is a time constant (for example, the RTT) of session i, and $\alpha_d > 1$ is the increase factor of the discrete-time algorithm. The above formulation is a continuous time interpolation of a multiplicative algorithm in which, for every RTT without congestion signals, the sender multiplies the window by a factor of α_d. This can be verified by substituting $t = n\tau_i$ in Equation (4.10). Since the sessions are homogeneous, we shall take $\alpha_1 = \alpha_2 = \alpha$. We assume that the two sessions react to congestion signals from the network. Unlike the model in [CJ89], the two sessions need not receive a congestion signal at the same time instant. Let $0 < \beta < 1$ be the decrease factor. Let t_n denote the time instant at which a congestion signal[11] is generated. Due to the possibility of asynchronous congestion signals, there are three possible rate vectors at t_n+, i.e., just after the congestion signal is sent. The possible rate vectors are given in Table 4.1. A 0 signal represents a no congestion signal whereas a 1 signal represents a congestion signal. On the reception of a congestion signal, a session instantaneously reduces its rate by a factor of β.

[11] We have assumed that the feedback is instantaneous.

4.5.1 The instantaneous rate ratio process

For two sessions we can rewrite Equation (4.7) as

$$F_* = \frac{1}{2} + \inf_{\mathsf{x}(0)} \lim_{T\to\infty} \frac{1}{T} \int_0^T \frac{\mathsf{x}_1(t)\mathsf{x}_2(t)}{\mathsf{x}_1(t)^2 + \mathsf{x}_2(t)^2} dt. \tag{4.12}$$

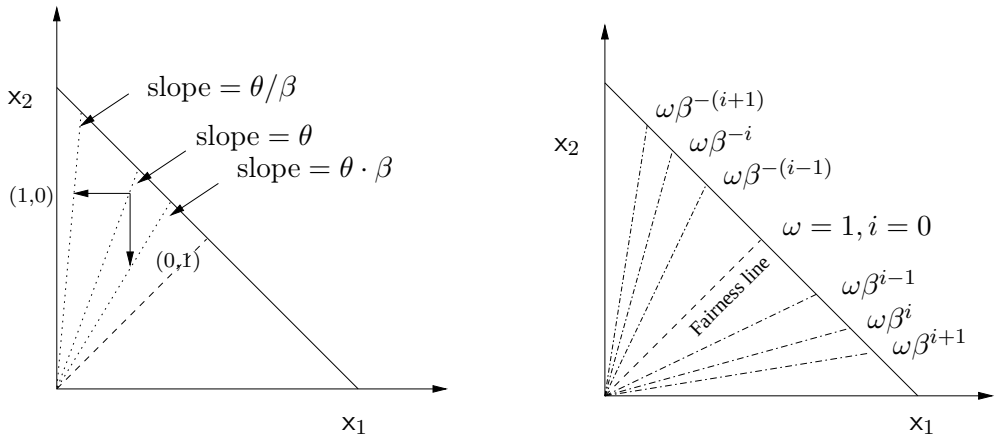
From the above equation we can infer that the fairness index can be studied as a function of

$$\theta(t) = \frac{\mathsf{x}_2(t)}{\mathsf{x}_1(t)} \tag{4.13}$$

rather than as function of the vector $\mathsf{x}(t)$. In terms of $\theta(t)$ we can rewrite the fairness index as

$$F_* = \frac{1}{2} + \inf_{\mathsf{x}(0)} \lim_{T\to\infty} \frac{1}{T} \int_0^T \frac{\theta(t)}{1+\theta(t)^2} dt. \tag{4.14}$$

Since the fairness index can be studied as a function of $\theta(t)$, we now study the behaviour of the process $\{\theta(t), t \geq 0\}$. We shall show that $\{\theta(t)\}$ is a Markov process with a *countable* state space. We shall then show that this process could be stable or unstable depending on the asynchronous loss process.



(a) Rate vector, x (b) State space, $\mathbb{S}$, of $\theta(t)$

Figure 4.4: Geometric interpretation.

We note that $\theta(t)$ is the slope of the rate vector, $\mathsf{x}(t)$. In Figure 4.4(a), we show the geometric interpretation of the response to the different congestion signals. If there were no congestion signals during the interval $(t, t+\tau)$ then the rate vector at time $(t+\tau)$ would be $\alpha^{\tau/\tau_0}(\mathsf{x}_1(t), \mathsf{x}_2(t))$. This vector will have a slope of $\theta(t)$. If a synchronous congestion signal were to be generated at time t, then the rate vector after the response, $\mathsf{x}(t+)$, would be $\beta(\mathsf{x}_1(t), \mathsf{x}_2(t))$. This vector will also have a slope of $\theta(t)$. Therefore, the rate vector remains along the same line as long as either there are no congestion signals or there are only synchronous congestion signals. However, the rate vector moves to the line with slope $\theta(t)/\beta$ when a congestion signal $(1,0)$ is generated. Similarly, the rate vector moves to the line with slope $\theta(t)\cdot\beta$ when a congestion signal $(0,1)$ is generated. Therefore, if there were only synchronous congestion signals (i.e., $(1,1)$ signals) then $\theta(t)$ would be equal to $\theta(0)$ for all t, and any initial unfair sharing would remain forever. Hence, in order to improve the fairness

index of systems with sessions using the MIMD algorithm, it is necessary to provide a stream of asynchronous congestion signals using, for example, some active queue management scheme.

Let t_n denote the time instant when the nth congestion signal is generated. Let $\theta(n) = \theta(t_n+)$ be the slope of the rate vector sampled just after the nth congestion signal is generated. From Figure 4.4(a), it can be seen that, given an initial slope of $\theta(0)$, the slope of the line along which the rate vector lies just after the nth control signal is generated can be written as

$$\theta(n) = \theta(0)\beta^i \tag{4.15}$$

for some $i \in \mathbb{Z}$. Then, for any given initial slope, $\theta(0)$, we can find a unique $\omega \in [\beta^{1/2}, \beta^{-1/2})$ and $j \in \mathbb{Z}$ such that

$$\theta(0) = \omega\beta^j. \tag{4.16}$$

For convenience, we shall define $\theta(n)$ in terms of ω and β. From the preceding two arguments, the state space of $\theta(n)$ can be represented by the countably infinite state space

$$\mathbb{S} = \{\omega\beta^i, \forall i \in \mathbb{Z}\}. \tag{4.17}$$

A geometric interpretation of $\mathbb{S}$ is shown in Figure 4.4(b). We note that the slope $\omega = 1, i = 0$ corresponds to the slope of the fairness line.

Let us define the process

$$\mathsf{s}(t) = \frac{\log[\theta(t)] - \log[\omega]}{\log[\beta]}, \quad \forall t. \tag{4.18}$$

With this definition, the state space of the process $\{\mathsf{s}(t)\}$ is the set of integers $\mathbb{Z}$. We note that the evolution of $\mathsf{s}(t)$ can be described by the equation

$$\mathsf{s}(t_n+) = \begin{cases} \mathsf{s}(t_n) + 1 & \text{if the } n\text{th congestion signal is } (0,1); \\ \mathsf{s}(t_n) - 1 & \text{if the } n\text{th congestion signal is } (1,0); \\ \mathsf{s}(t_n) & \text{otherwise.} \end{cases} \tag{4.19}$$

We shall embed this process at instants at which a congestion signal is generated.Let $s(n) = \mathsf{s}(t_n)$ denote the embedded process. We will show that $s(n)$ is a Markov chain, and that it is stable or unstable depending upon the asynchronous loss process.

4.5.2 The modified loss process

A congestion signal is generated either when the capacity is achieved or can be generated by the network before the capacity is achieved. We have assumed that when the capacity is achieved a synchronous congestion signal is generated. Since a synchronous congestion signal does not affect the evolution of the process $\{\theta(t)\}$, we can ignore the congestion signals which are generated when capacity is achieved. Hence, we can analyse the fairness index by assuming the loss process to be the one in which the congestion signals are generated by the network. Hence, the loss process can be considered to be a Poisson process of intensity λ. In the rest of this section, we shall refer to the network generated congestion signals as "congestion signals".

Remark 4.5.2 *We are studying the behaviour of the fairness index as a function of the rate ratio process and not as a function of the individual rates,* $\mathsf{x}(t)$*, themselves. Hence, the individual rates are not modelled separately. However, due to the capacity constraints,* $\mathsf{x}_i(t)$ *affects the time interval between two congestion signals, i.e.,*

$$t_{n+1} \leq t_n + \frac{1}{\alpha} \log \left[\frac{C}{x_1(n) + x_2(n)} \right]. \tag{4.20}$$

Since we can ignore the congestion signals generated due to capacity constraints, we can consider the fairness index to be a function of $\{\theta(t)\}$ *and the network generated congestion signals of intensity* λ*. Hence, the capacity of the link does not affect the fairness index.*

4.5.3 The rate-independent loss model

First we shall study the evolution of the process $\{s(n), n \geq 0\}$ when the asynchronous losses are rate-independent. We note that $\mathsf{s}(t)$ remains constant between two asynchronous loss instants. From the description of the loss model in Equation (4.5) along with the evolution of $\mathsf{s}(t)$ (Equation (4.19)), the transition probabilities for $s(n)$ are given by

$$\text{Prob}(s(n+1) = j | s(n) = i) = \begin{cases} \frac{\epsilon}{2} & \text{if} \quad j = i-1; \\ \frac{\epsilon}{2} & \text{if} \quad j = i+1; \\ 1-\epsilon & \text{if} \quad j = i. \end{cases} \tag{4.21}$$

From the transition probabilities, we can infer that the process $\{s(n)\}$ forms a Markov chain with an irreducible state space, $\mathbb{Z}$.

Proposition 4.5.1 *For* $0 < \epsilon < 1$*, the Markov chain with transition probabilities defined in Equation (4.21) is null recurrent.*

Proof. For this Markov chain to be null recurrent, we use a theorem from [FMM95], which we also state as Theorem C.0.1 in Appendix C.

Let $f(x) = x^2$, $\psi(x) = |x|$, $A = \{0\}$, $\gamma = 1$ and $\alpha = 2$. Conditions 1, 2 and 3 of Theorem C.0.1 are satisfied with these assumptions. For condition 4(a),

$$\Delta f_i = \frac{\epsilon}{2}((x+1)^2 - x^2) + \frac{\epsilon}{2}((x-1)^2 - x^2) = \epsilon > 0.$$

For condition 4(b),

$$\Delta \psi_i = \frac{\epsilon}{2}((|x|+1) - |x|) + \frac{\epsilon}{2}((|x|-1) - |x|) = 0.$$

For condition 4(c),

$$\begin{aligned} E[|\psi(s(n+1)) - \psi(s(n))|^2 \, | s(n) = x] &= \frac{\epsilon}{2}((|x|+1) - |x|)^2 + \frac{\epsilon}{2}((|x|-1) - |x|)^2 \\ &= \epsilon < \infty. \end{aligned}$$

Hence, the Markov chain s is null recurrent. ∎

Since s is null recurrent, and the discrete state space has two accumulation points, $-\infty$ (corresponding to $\mathsf{x}_1 \to 0$) and ∞ (corresponding to $\mathsf{x}_2 \to 0$), the probability of being in any small

vicinity of each point is 0.5. The mean time to go from one extreme to another will be ∞ and, hence, the rate of one of the connections will be zero. This suggests that rate independent losses are not sufficient to improve the fairness index.

Remark 4.5.3 *We are taking the limit* $\mathsf{x}_i \to 0$ *since the rate* $\mathsf{x}_i(t)$ *of an MIMD session takes values in the set* $(0, \infty)$ *if* $\mathsf{x}_i(0) > 0$*, and the rate is zero if* $\mathsf{x}_i(0) = 0$*. Also, we have assumed that the minimum possible rate available to any session is not bounded away from zero. If the minimum possible rate is bounded away from zero, then the state space will be finite. In this case the Markov chain,* $s(n)$*, will be positive recurrent. The steady state distribution for this Markov chain will be uniform, and the exact fairness index could be computed using*

$$F_{ri} = \frac{1}{2} + \frac{1}{2L+1} \sum_{i=-L}^{L} \frac{\omega\beta^i}{1+\omega\beta^{2i}}, \tag{4.22}$$

where we have assumed that the state space of $s(n)$ *has* $2L+1$ *states that are symmetrically placed about state* 0*. We can also write this equation in the form*

$$F_{ri} = \frac{1}{2} + \frac{1}{2L+1} \left[\frac{\omega}{1+\omega} + \sum_{i=1}^{L} \left(\frac{\omega\beta^i}{1+\omega\beta^{2i}} + \frac{\omega\beta^i}{\beta^{2i}+\omega} \right) \right]. \tag{4.23}$$

On substituting for $\beta = 0$ *in the denominator of the RHS, we obtain*

$$\frac{1}{2} + \frac{1}{2L+1} \left[\frac{\omega}{1+\omega} + (\omega+1)\frac{\beta - \beta^{L+1}}{1-\beta} \right], \tag{4.24}$$

which is an upper bound on F_{ri}*. As* L *tends to* ∞ *the upper bound goes to* 0.5*, thereby resulting in a minimal value for the fairness index.*

Remark 4.5.4 *Since the loss process is assumed to be a Poisson process, the continuous time Markov process,* $\{s(t)\}$*, can also be considered as a piecewise deterministic Markov process (see, for example, [Dav93]). Although the embedded process* $\{s(n)\}$ *is null recurrent, there may exist an invariant measure for the process* $\{s(t)\}$ *[DC99, Example 2]. However, in our model the jump rate is positive and bounded, hence there is a one-to-one mapping between the invariant probability measure of the embedded process,* $\{s(n)\}$*, and the invariant probability measure of* $\{s(t)\}$ *[DC99].*

4.5.4 The rate-dependent loss model

Next, we study the behaviour of $\{s(n)\}$ when the probability that a session receives the nth congestion signal depends on its sending rate, $s(n)$. The loss model is described by Equation (4.4).

Proposition 4.5.2 *The process* $\{s(n), n \geq 0\}$ *is a discrete state-space Markov chain with transition probabilities given by*

$$Prob(s(n+1) = j | s(n) = i) = \begin{cases} \epsilon \frac{\omega\beta^i}{1+\omega\beta^i} & \text{if} \quad j = i+1; \\ \epsilon \frac{1}{1+\omega\beta^i} & \text{if} \quad j = i-1; \\ 1-\epsilon & \text{if} \quad j = i. \end{cases} \tag{4.25}$$

We note that the state space $\mathbb{Z}$ is irreducible for the $\{s(n)\}$. We now show that the limiting process $\lim_{n\to\infty} s(n)$ converges to a stationary and ergodic process with a unique invariant probability measure. The existence of the limiting distribution can be ensured by proving that the Markov chain $\{s(n)\}$ is positive recurrent.

Proposition 4.5.3 *For $0 < \beta < 1$ and $0 < \epsilon \leq 1$, the Markov chain, $\{s(n)\}$, defined in Proposition 4.5.2 is positive recurrent.*

Proof. A Markov chain is positive recurrent if it satisfies the Foster's criterion [FMM95] which we state as Theorem C.0.2 in Appendix C.

Let $f(i) = |i|, i \in \mathbb{Z}$ and $A = \{0\}$. Let $\Delta f_i := E[f(s(n+1)) - f(s(n))|s(n) = i]$. First, we show that condition (C.1a) is satisfied. We note that $\omega \in [\beta^{1/2}, \beta^{-1/2})$. For $i < 0$, from Equation (4.25), we have

$$\begin{aligned}
\Delta f_i &= \epsilon\frac{\omega\beta^i}{1+\omega\beta^i}(|i|-1) + \epsilon\frac{1}{1+\omega\beta^i}(|i|+1) + (1-\epsilon)|i| - |i| && (4.26)\\
&= \epsilon\frac{1-\omega\beta^i}{1+\omega\beta^i} && (4.27)\\
&= -\epsilon\left(1 - 2\frac{\beta^{|i|}}{\beta^{|i|}+\omega}\right). && (4.28)\\
&\leq -\epsilon\left(1 - 2\frac{\beta}{\beta+\omega}\right). && (4.29)
\end{aligned}$$

which is strictly negative for any $\beta \in (0,1)$ and $\epsilon \in (0,1]$. Similarly, for $i > 0$, we can show that

$$\Delta f_i \leq -\epsilon\left(\frac{2}{\omega\beta+1} - 1\right). \tag{4.30}$$

Therefore, condition (C.1a) is satisfied. To check for (C.1b), for $i = 0$ we have

$$E[f(s(n+1))|s(n) = i] = \epsilon\frac{1}{1+\omega}|-1| + \epsilon\frac{\omega}{1+\omega}|1| = \epsilon < \infty.$$

Therefore, the Markov chain $\{s(n)\}$ satisfies the conditions of Theorem C.0.2 and, hence, is positive recurrent. ∎

Let $\tilde{s}$ and $\tilde{\theta}$ denote respectively the stationary ergodic process to which the Markov chain $\{s_n\}$ and the process $\{\theta(t)\}$ converge to when $n \to \infty$. We note that the fairness index in Equation (4.14) can be rewritten as

$$F_{rd} = \frac{1}{2} + \mathrm{E}\left[\frac{\tilde{\theta}}{1+\tilde{\theta}^2}\right]. \tag{4.31}$$

Since the loss process is assumed to be a Poisson process, we can use the PASTA property and Equation (4.18) to evaluate the fairness index as

$$F_{rd} = \frac{1}{2} + \mathrm{E}\left[\frac{\omega\beta^{\tilde{s}}}{1+\omega^2\beta^{2\tilde{s}}}\right]. \tag{4.32}$$

We shall now find the distribution of $\tilde{s}$ and also the mean first passage time to the state 0 starting from a random state. We note that the state 0 corresponds to the state closest to the fairness line, i.e., to the the state $\theta(t) = \omega$. Therefore, the mean first passage time from a random state to the state 0 gives an indication of the mean time for the rate vector to get closest to the fairness line. We again note that if $\omega \neq 1$, then the process $\{\theta(t)\}$ cannot be on the fairness line but can get close to the fairness line by reaching ω.

4.5.5 Steady state distribution

Let π_i be the steady state probability of $\tilde{s}$ being in state i.

Proposition 4.5.4 *The probability* $\pi(i) \stackrel{\Delta}{=} Prob(\tilde{s} = i)$ *can be obtained as*

$$\pi(i) = \frac{\pi(0)}{1+\omega}\omega^i \beta^{i(i-1)/2}(1+\omega\beta^i), \quad \forall i, \tag{4.33}$$

$$where \ \pi(0) = \frac{1+\omega}{\sum_{i=-\infty}^{\infty} \omega^i \beta^{i(i-1)/2} + \omega^{i+1}\beta^{(i+1)i/2}}. \tag{4.34}$$

Proof. From Equation (4.25), in steady state $\pi(i)$ satisfies the balance equation

$$\pi(i) = (1-\epsilon)\pi(i) + \epsilon\frac{\omega\beta^{i-1}}{1+\omega\beta^{i-1}}\pi(i-1) + \epsilon\frac{1}{1+\omega\beta^{i+1}}\pi(i+1). \tag{4.35}$$

Or,

$$\pi(i) = \frac{\omega\beta^{i-1}}{1+\omega\beta^{i-1}}\pi(i-1) + \frac{1}{1+\omega\beta^{i+1}}\pi(i+1). \tag{4.36}$$

Now, we verify that $\pi(i)$ as defined in Equation (4.33) satisfies Equation (4.36). On substituting for $\pi(\cdot)$ from Equation (4.33) in right hand side of Equation (4.36), we obtain

$$\frac{\omega\beta^{i-1}}{1+\omega\beta^{i-1}}\pi(i-1) + \frac{1}{1+\omega\beta^{i+1}}\pi(i+1) = \frac{\pi(0)}{1+\omega}(\omega^i\beta^{i-1}\beta^{(i-1)(i-2)/2} + \omega^{i+1}\beta^{(i+1)i/2}) \tag{4.37}$$

$$= \frac{\pi(0)}{1+\omega}\omega^i\beta^{i(i-1)/2}(1+\omega\beta^i) \tag{4.38}$$

$$= \pi(i). \tag{4.39}$$

The probability $\pi(0)$ can be obtained by using the identity

$$\sum_{i=-\infty}^{\infty} \pi(i) = 1. \tag{4.40}$$

■

Since $\beta^i \to 0$ as $i \to \infty$, the tail of $\pi(i)$ can be seen to decrease as β^{i^2}, which is a very fast decrease.

Remark 4.5.5 *Since the arrivals are Poisson with constant intensity, we can use PASTA and infer that the steady state distribution of the embedded chain,* $\theta(n)$*, are also the time average distribution for the process* $\theta(t)$*. We note that the steady state probabilities are independent of* λ *and* ϵ*.*

Proposition 4.5.5 *The jth moment of the rate ratio process*

$$E[\tilde{\theta}^j] = \frac{1+\beta^{-j}}{2}\beta^{-j(j-1)/2}. \tag{4.41}$$

Proof. To prove this result we shall use the Jacobi triple product identity [HW79]

$$\vartheta_2(z,q) \stackrel{\Delta}{=} \sum_{i=-\infty}^{\infty} z^i q^{i(i-1)/2} = \prod_{n=1}^{\infty}(1-q^{n-1})(1+zq^{n-1})\left(1+\frac{q^n}{z}\right). \tag{4.42}$$

From Equation (3.13) and Equation (4.42), we can write

$$\pi(0) = \frac{1+\omega}{2\vartheta_2(\omega,\beta)}. \tag{4.43}$$

From Equation (4.18), we note that $\text{Prob}(\tilde{\theta} = \omega\beta^i) = \text{Prob}(\tilde{s} = i)$. The LHS of Equation (4.41) can be written as

$$\begin{aligned}
\text{E}[\tilde{\theta}^j] &= \sum_{i=-\infty}^{\infty}(\omega\beta^i)^j\pi(i) && (4.44)\\
&= \sum_{i=-\infty}^{\infty}\frac{\pi(0)}{1+\omega}(\omega^i\beta^{i(i-1)/2} + \omega^{i+1}\beta^{(i+1)i/2})\omega^j\beta^{ij} && (4.45)\\
&= \sum_{i=-\infty}^{\infty}\frac{\pi(0)\omega^j}{1+\omega}(\omega^i\beta^{i(i-1)/2}\beta^{ij} + \omega^{i+1}\beta^{(i+1)i/2}\beta^{ij}). && (4.46)
\end{aligned}$$

We shall now evaluate each sum on the RHS separately. The first sum

$$\begin{aligned}
\sum_{i=-\infty}^{\infty}\omega^i\beta^{i(i-1)/2}\beta^{ij} &= \sum_{i=-\infty}^{\infty}(\omega\beta^j)^i\beta^{i(i-1)/2} && (4.47)\\
&= \vartheta_2(\omega\beta^j,\beta). && (4.48)
\end{aligned}$$

From Equation (4.42)

$$\begin{aligned}
\vartheta_2(\omega\beta^j,\beta) &= \prod_{n=1}^{\infty}(1-\beta^{n-1})(1+\omega\beta^{n-1+j})\left(1+\frac{\beta^{n-j}}{\omega}\right) && (4.49)\\
&= \prod_{n=1}^{\infty}(1-\beta^{n-1})(1+\omega\beta^{n-1})\left(1+\frac{\beta^n}{\omega}\right)\prod_{k=0}^{j-1}\frac{1+\frac{\beta^{-k}}{\omega}}{1+\omega\beta^k} && (4.50)\\
&= \vartheta_2(\omega,\beta)\omega^{-j}\beta^{-j(j-1)/2}. && (4.51)
\end{aligned}$$

Similarly, the second sum on the RHS

$$\begin{aligned}
\sum_{i=-\infty}^{\infty}\omega^{i+1}\beta^{i(i+1)/2}\beta^{ij} &= \beta^{-j}\sum_{i=-\infty}^{\infty}(\omega\beta^j)^{i+1}\beta^{i(i+1)/2} && (4.52)\\
&= \beta^{-j}\vartheta_2(\omega\beta^j,\beta) && (4.53)\\
&= \beta^{-j}\vartheta_2(\omega,\beta)\omega^{-j}\beta^{-j(j-1)/2}. && (4.54)
\end{aligned}$$

Therefore Equation (4.46) can be written as

$$\mathrm{E}[\tilde{\theta}^j] = \frac{\pi(0)}{1+\omega}\vartheta_2(\omega,\beta)\beta^{-j(j-1)/2}(1+\beta^{-j}) \tag{4.55}$$

On substituting for $\pi(0)$ from Equation (4.43) into Equation (4.55), we obtain

$$\mathrm{E}[\tilde{\theta}^j] = \frac{1+\beta^{-j}}{2}\beta^{-j(j-1)/2}. \tag{4.56}$$

■

We note that the moments of the rate ratio process are independent of ω, and, hence, are independent of $\mathsf{x}(0)$.

From Equation (4.34), we can infer that

$$\lim_{\beta\downarrow 0}\pi(0) = \frac{1}{2}, \tag{4.57}$$

$$\text{and} \quad \lim_{\beta\uparrow 1}\pi(0) = 0. \tag{4.58}$$

The fairness index, F_{rd}, can be computed numerically using Equation (4.7) and Equation (4.33). In Table 4.2 we give the fairness index for different values of β for $\omega = 1$. In the numerical study

Table 4.2: Fairness index for different values of β. $\omega = 1$.

β	0.95	0.875	0.75	0.6	0.5	0.1
F_{rd}	0.987	0.97	0.942	0.91	0.88	0.777

we observed that the fairness index was independent of ω as well. Also, it was observed that

$$\lim_{\beta\downarrow 0}F_{rd} = \frac{3}{4}, \tag{4.59}$$

$$\text{and} \quad \lim_{\beta\uparrow 1}F_{rd} = 1. \tag{4.60}$$

We see that $\pi(0)$ and F_{rd} are respectively decreasing and increasing functions of β. Since $\pi(0)$ is also the average time spent in state 0, i.e., the state closest to the fairness line, we can also consider $\pi(0)$ to be an indicator of fairness. The maximization of $\pi(0)$ with respect to β will lead to a minima of F_{rd} and vice versa. Hence, the choice of the parameter β will be depend on the fairness criterion that we choose to maximize.

4.5.6 Convergence to steady state distribution

The second largest eigenvalue of a matrix gives the rate of convergence to the steady state distribution. Therefore, we can get an indication of the rate of convergence of the Markov chain $\tilde{s}$ by

looking at the eigenvalues of its transition probability matrix,

$$P = \begin{array}{c} \\ \vdots \\ -1 \\ 0 \\ 1 \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} \dots & -1 & 0 & 1 & \dots \end{array} \\ \begin{pmatrix} \ddots & \vdots & \vdots & \vdots & \dots \\ \dots & 1-\epsilon & \epsilon q_{-1} & 0 & \dots \\ \dots & \epsilon(1-q_0) & 1-\epsilon & \epsilon q_0 & \dots \\ \dots & 0 & \epsilon(1-q_1) & 1-\epsilon & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array} \tag{4.61}$$

where $q_i = \frac{\omega\beta^i}{1+\omega\beta^i}$. Let ζ_i denote the ith eigenvalue of P such that $\zeta_i \geq \zeta_j$ for $i < j$, and $\zeta_0 = 1$. We can obtain the following lower bound on ζ_i.

Proposition 4.5.6 *For $i > 0$,*

$$1 - \epsilon \leq \zeta_i < 1. \tag{4.62}$$

Proof. Since $\tilde{s}$ is positive recurrent and irreducible, the multiplicity of eigenvalues at 1 is 1. Therefore, $\zeta_i < 1$ for $i > 0$. We can rewrite P as

$$P = (1-\epsilon)I + \epsilon A \tag{4.63}$$

where I is the identity matrix and A is a transition matrix of a birth-death process with up transition probability q_i and down transition probability $1 - q_i$. The matrix A is a stochastic matrix and, therefore, all its eigenvalues belong to the interval $[0, 1]$. Let μ_i be the ith eigenvalue of A and let v_i be the corresponding left eigenvector. Then, from Equation (4.63) we get

$$v_i P = (1-\epsilon)v_i + \epsilon v_i A = ((1-\epsilon) + \epsilon\mu_i)v_i.$$

Therefore, v_i is also the left eigenvector of P, and the corresponding eigenvalue is $(1-\epsilon) + \epsilon\mu_i$. Since $\mu_i \geq 0$, we get the inequality $\zeta_i \geq 1 - \epsilon$. ■

Therefore, $1 - \epsilon$ gives a lower bound on the rate of convergence of the Markov chain to the steady state. We note that $1 - \epsilon$ is the probability of a synchronized network generated congestion signal. Hence, if the probability of a synchronous loss is large, then the convergence to the steady state will be slow.

4.5.7 Mean first passage time

In this section we compute the mean first passage time to the state 0 starting from a given state. This gives us an estimate of the first time the rate vector reaches the fairness line starting from a given initial state. The Markov chain $\{s(n)\}$ is a birth-death process. If the initial state is positive, the Markov chain will stay in the set of positive states before visiting state 0. Similarly, if the initial state is negative, the Markov chain will stay in the set of negative states before visiting state 0. Therefore, we can obtain the mean first passage time to state 0 by considering two separate chains, one for the set of positive states and the other for the set of negative states.

Let m_i denote the mean first passage time from i to state 0. First, we shall consider the case $i > 0$.

Proposition 4.5.7 *The mean first passage times, $m_i, i \geq 1$ can be obtained by using following recursion: $m_0 = 0$,*

$$m_{i+1} = \frac{\epsilon m_i - \epsilon(1-q_i)m_{i-1} - 1}{\epsilon q_i}, \forall i > 0, \tag{4.64}$$

$$\textit{with } m_1 = \frac{1-\pi_-}{\epsilon\pi(0)q_0}, \tag{4.65}$$

$$\textit{and } m_0 = 0, \tag{4.66}$$

where $\pi_- = \sum_{i\leq 0}\pi(i)$.

Proof. The mean first passage time, m_{ij}, of a Markov chain from state i to j can be obtained by using the recursion [KSK76]

$$m_{ij} = 1 + \sum_{k\neq j} p_{ik}m_{kj}, \tag{4.67}$$

where p_{ik} is the transition probability from i to j. From the transition probability matrix given in Equation (4.61), we obtain the recursion

$$m_{i+1} = \frac{\epsilon m_i - \epsilon(1-q_i)m_{i-1} - 1}{\epsilon q_i}. \tag{4.68}$$

To obtain the m_1, we consider the set of states $\mathbb{Z}_- = \{\ldots, -2, -1, 0\}$. The probability of being in these states is π_-. Hence, the mean recurrence time for $\mathbb{Z}_-$ is $1/\pi_-$. The mean recurrence time, r_-, can also be expressed as

$$r_- = 1 + p_- m_1, \tag{4.69}$$

where p_- is the probability of going from $\mathbb{Z}_-$ to state 1. Since the chain $\{s_n\}$ can go from $\mathbb{Z}_-$ to state 1 only through state 0,

$$p_- = \frac{\pi(0)}{\pi_-}\epsilon q_0. \tag{4.70}$$

Hence, we have

$$m_1 = \frac{1-\pi_-}{\epsilon\pi(0)q_0}. \tag{4.71}$$

■

Similarly, we can obtain the recurrence relation for mean first passage time from state i, $i \leq -1$ as

$$m_{i-1} = \frac{\epsilon m_i - \epsilon q_i m_{i+1} - 1}{\epsilon(1-q_i)}, \forall i < 0, \tag{4.72}$$

$$\text{with } m_{-1} = \frac{1-\pi_+}{\epsilon\pi(0)(1-q_0)}, \tag{4.73}$$

$$\text{and } m_0 = 0, \tag{4.74}$$

where $\pi_+ = \sum_{i\geq 0}\pi(i)$.

The mean first passage time to state 0 is in the units of steps to reach state 0. In order to get the absolute time we have to multiply this by the mean time between each step, i.e., by λ^{-1}. We note that the steady state probabilities are independent of ϵ whereas the mean first passage times are inversely proportional to ϵ.

4.6 Intra-algorithm fairness of the MIMD algorithm: heterogeneous sessions

In this section we shall assume that the two sessions are heterogeneous. The heterogeneity could be either due to unequal increase parameters or due to unequal time constants or both. From Equation (4.10), the rate evolution for session i in the absence of control signals is given by

$$\begin{aligned} \mathsf{x}_i(t+\tau) &= \exp(\alpha_i \tau)\mathsf{x}_i(t), i = 1, 2, & (4.75)\\ \text{where } \alpha_i &= \log[\alpha_{di}]/\tau_i, & (4.76) \end{aligned}$$

and α_{di} and τ_i are respectively the discrete-time increase parameter and the RTT of session i.

For heterogeneous sessions, we again define the process $\mathsf{s}(t)$ as

$$\mathsf{s}(t) = \log\left[\frac{\mathsf{x}_2(t)}{\mathsf{x}_1(t)}\right]. \tag{4.77}$$

We note that $\mathsf{s}(t)$ is the logarithm of the rate ratio process.

In the absence of asynchronous congestion signals, the process $\{\mathsf{s}(t)\}$ evolves as

$$\mathsf{s}(t+\tau) = \mathsf{s}(t) + \kappa\tau, \tag{4.78}$$

where $\kappa = \alpha_2 - \alpha_1$. Let t_n denote the time instant when the nth asynchronous congestion signal is generated by the network. The logarithm of the rate ratio just after this congestion signal is given as

$$\mathsf{s}(t_n+) = \begin{cases} \mathsf{s}(t_n) + \log[\beta] & \text{signal is (0,1);} \\ \mathsf{s}(t_n) - \log[\beta] & \text{signal is (1,0).} \end{cases} \tag{4.79}$$

The evolution in time of $\mathsf{s}(t)$ is shown in Figure 4.5. in which we have assumed that κ is positive. For simplicity let us assume that the heterogeneity is only due to unequal RTTs. For

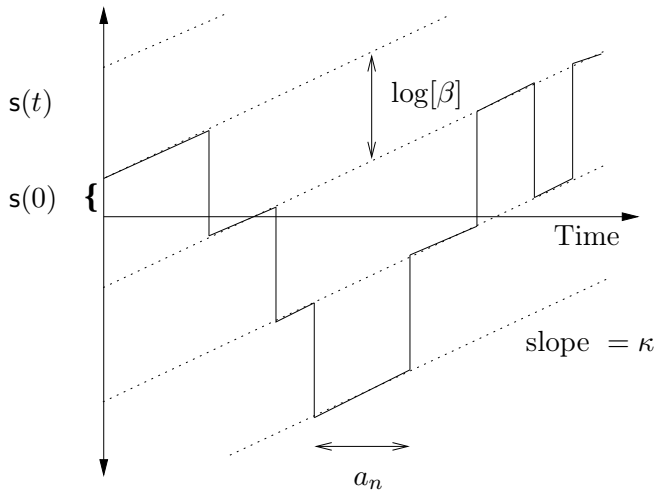


Figure 4.5: Evolution in time of $\mathsf{s}(t)$. $\tau_2 < \tau_1$.

$\kappa > 0$ (i.e., $\tau_2 < \tau_2$), from Equation (4.78) and Equation (4.79), it can be seen that if there were only synchronous congestion signals then there would be a drift in time towards ∞, i.e.,

$\lim_{t\to\infty} \mathsf{s}(t) = +\infty$. This suggests that the rate for session 1 would approach zero. Similarly, if $\kappa < 0$ (i.e., $\tau_2 > \tau_1$) and there were to be only synchronous congestion signals, then there would be a drift in time towards $-\infty$, i.e., $\lim_{t\to\infty} \mathsf{s}(t) = -\infty$. This suggests that the rate for session 2 would approach zero. It can also be seen that if $\kappa = 0$, i.e., $\tau_1 = \tau_2$, the dotted lines would be parallel to the time axis. In this case $\mathsf{s}(t)$ would remain constant in the absence of asynchronous losses, which was also observed in the previous section. We also observed in the previous section that the losses needed to be *rate-dependent* in order to improve the fairness index. Hence, we conclude that even for the system with heterogeneous sessions rate-dependent congestion signals are necessary in order to improve the fairness index.

We note that as was mentioned in Section 4.5.2, we can consider only the congestion signals to generated by the network. We also observed in Proposition 4.5.3 that any $\epsilon > 0$ was sufficient to ensure stability and that the value of ϵ does not affect the stability. We shall therefore assume that the network generates only asynchronous congestion signals, i.e., $\epsilon = 1$. In the rest of this section, when we use the term "congestion signal" we shall mean "a rate-dependent congestion signal generated by the network".

Let $\{a_n, n \geq 0\}$ denote the time interval between the generation of the nth and the $(n+1)$th congestion signal. We shall assume that the sequence $\{a_n,\ n \geq 0\}$ is an i.i.d. sequence with distribution function[12] A and intensity λ. We shall assume that A has a density function which is non-increasing. Let $\{\mathsf{s}(n)\} = \{\mathsf{s}(t_n-)\}$ denote the process $\{\mathsf{s}(t)\}$ embedded just before the the nth loss instant.

Proposition 4.6.1 *The process* $\{\mathsf{s}(n),\ n \geq 0\}$ *is a Markov chain with state space* $\mathbb{R}$*, and it follows the recursive equation*

$$\mathsf{s}(n+1) = \mathsf{s}(n) + \kappa a_n + d_n, \tag{4.80}$$

where d_n *is defined as*

$$d_n = \begin{cases} +\log[\beta] & w.p.\ \frac{\exp(\mathsf{s}(n))}{1+\exp(\mathsf{s}(n))}; \\ -\log[\beta] & w.p.\ \frac{1}{1+\exp(\mathsf{s}(n))}. \end{cases} \tag{4.81}$$

Proof. The formulation follows from Equation (4.78), Equation (4.79) and Equation (4.4). ∎

We shall now provide a condition on the intensity λ such that the Markov chain $\{\mathsf{s}(n)\}$ is positive recurrent. By showing that the Markov chain is positive recurrent, we can show that there will not be a drift towards $+\infty$ or $-\infty$. This would then result in an improvement in the fairness index. In the rest of this section we shall assume that $\kappa > 0$. We note that when $\kappa < 0$ we can define $\mathsf{s}(t) = \log\left[\frac{\mathsf{x}_1(t)}{\mathsf{x}_2(t)}\right]$, and then the analysis would be similar.

4.6.1 Stability

Let us define $\hat{a}_n = \kappa a_n$ such that the distribution function of $\hat{a}_n$ is $\hat{A}$ and the intensity of $\hat{a}_n$ is $\hat{\lambda} = \kappa^{-1}\lambda$. Let $b = -\log[\beta]$. We note that b is a positive number because β is less than 1.

Proposition 4.6.2 *The Markov chain* $\{\mathsf{s}(n)\}$ *defined by Equation (4.80) is positive recurrent if*

$$\hat{\lambda}^{-1} < b. \tag{4.82}$$

[12] We relax the Poissonian assumption just for this section.

Proof. We note that the transition probabilities of $\{\mathsf{s}(n)\}$ are time homogeneous. In order to show the positive recurrence of $\{\mathsf{s}(n)\}$, we can use the a theorem from [MT93] which we state as Theorem C.0.3 in Appendix C.

In order to check for the drift condition of this theorem, we consider $V(y) = |y|$, and the set $W = [-b, b]$. The LHS of Equation (C.3)) becomes

$$\begin{aligned}
\Delta V(x) + |x| &= \int_{y\in\mathbb{R}} \frac{1}{1+e^x}|y|d\hat{A}(y-(x+b)) + \int_{y\in\mathbb{R}} \frac{e^x}{1+e^x}|y|d\hat{A}(y-(x-b)) \\
&= \int_{y=(x+b)}^{\infty} \frac{1}{1+e^x}|y|d\hat{A}(y-(x+b)) + \int_{y=(x-b)}^{\infty} \frac{e^x}{1+e^x}|y|d\hat{A}(y-(x-b)) \\
&= \int_{0}^{\infty} \frac{1}{1+e^x}|y+(x+b)|d\hat{A}(y)) + \int_{0}^{\infty} \frac{e^x}{1+e^x}|y+(x-b)|d\hat{A}(y) \\
&\leq \int_{0}^{\infty} \frac{1}{1+e^x}(|y|+|x+b|)d\hat{A}(y)) + \int_{0}^{\infty} \frac{e^x}{1+e^x}(|y|+|x-b|)d\hat{A}(y) \\
\Delta V(x) + |x| &\leq \hat{\lambda}^{-1} + \int_{0}^{\infty} \frac{1}{1+e^x}|x+b|d\hat{A}(y) + \int_{0}^{\infty} \frac{e^x}{1+e^x}|x-b|d\hat{A}(y). \qquad (4.83)
\end{aligned}$$

For $x \in W$, Equation (4.83) can be rewritten as

$$\begin{aligned}
\Delta V(x) + |x| &\leq \hat{\lambda}^{-1} + \int_{0}^{\infty} \frac{1}{1+e^x}(|x|+|b|)d\hat{A}(y) + \int_{0}^{\infty} \frac{e^x}{1+e^x}(|x|+|b|)d\hat{A}(y) \\
\Delta V(x) &\leq \hat{\lambda}^{-1} + b \ < \infty.
\end{aligned}$$

For $x \in W^c$, Equation (4.83) can be rewritten as

$$\begin{aligned}
\Delta V(x) + |x| &\leq \hat{\lambda}^{-1} + \int_{0}^{\infty} \frac{1}{1+e^{|x|}}(|x|+b)d\hat{A}(y) + \int_{0}^{\infty} \frac{e^{|x|}}{1+e^{|x|}}(|x|-b)d\hat{A}(y) \\
\Delta V(x) &\leq \hat{\lambda}^{-1} + \frac{1-e^{|x|}}{1+e^{|x|}}b.
\end{aligned}$$

Let x^* be the value of x for which

$$\Delta V(x) \leq \hat{\lambda}^{-1} + \frac{1-e^x}{1+e^x}b = -\mu. \qquad (4.84)$$

Then, $x^* = \log \frac{b+\hat{\lambda}^{-1}+\mu}{b-(\hat{\lambda}^{-1}+\mu)}$. For $b > \hat{\lambda}^{-1} + \mu$, there exits a $x^* \in \mathbb{R}$ for which the $\Delta V(x) \leq -\mu$. If x^* is less than b then the drift condition is satisfied for the $W = [-b, b]$. However, if x^* is greater than b then we can consider the set $W = [-b, x^*]$. Hence, for $W = [-b, \max(b, x^*)]$, the drift condition (C.3) is satisfied.

Lemma 4.6.1 *For any d such that $-b \leq d < \infty$, the set $W = [-b, d]$ is a "small" set.*

Proof. For $x \in W$,

$$\begin{aligned}
P(x, B) &= \frac{1}{1+e^x} \int\limits_{y\in B} d\hat{A}(y-(x+b)) + \frac{e^x}{1+e^x} \int\limits_{y\in B} d\hat{A}(y-(x-b)) \\
&\geq \frac{e^x}{1+e^x} \int\limits_{y\in B} d\hat{A}(y+b-x). \qquad (4.85)
\end{aligned}$$

Since the density function of $\hat{A}$ was assumed to be a non-increasing function, then $\int_{y\in B} d\hat{A}(y+b-x)$ is a non-decreasing function in x. Also, $\frac{e^x}{1+e^x}$ is an increasing function in x. Since $x \geq -b$, we can rewrite (4.85) as

$$\begin{aligned} P(x,B) &\geq \frac{e^{-b}}{1+e^{-b}} \int_{y\in B} d\hat{A}(y+2b) \\ &\geq \phi(B), \end{aligned} \tag{4.86}$$

where $\phi(B) := \frac{e^{-b}}{1+e^{-b}} \int\limits_{y\in B} d\hat{A}(y+2b)$. Since there exists a measure ϕ such that $P(x,B) \geq \phi(B), \quad x \in W, B \in \mathbb{B}(\mathbb{R})$, the closed and bounded set $W = [-b,d]$ is a small set. ■

It follows from Lemma 4.6.1 that W is indeed a small set. Since W is a "small set" when $b > \hat{\lambda}$, from Theorem C.0.3 we can conclude that, for $b > \hat{\lambda}$, the Markov chain $\{\mathsf{s}(n)\}$ is positive recurrent. ■

We note that $b > \hat{\lambda}^{-1}$ is a sufficient condition for positive recurrence. Let us consider a modified continuous time process, $\{\hat{\mathsf{s}}(t)\}$, in which downward jumps occur with probability one whenever a congestion signal is generated. We note that in the $\{\mathsf{s}(t)\}$ process both upward and downward jumps occur according to Equation (4.80). For a given sequence $\{a_n, n \geq 0\}$, we see that at any t, $\mathsf{s}(t) \geq \hat{\mathsf{s}}(t)$. Since the process $\{\hat{\mathsf{s}}(t)\}$ drifts towards $+\infty$ if $\lambda < \kappa/b$, we can say that $\lambda < \kappa/b$ is a sufficient condition for transience of the process $\{\mathsf{s}(t)\}$.

From Proposition 4.6.2, we can say that a sufficient condition for improving the fairness index in a system with heterogeneous sessions is

$$\lambda > \frac{\alpha_2 - \alpha_1}{|\log[\beta]|}. \tag{4.87}$$

4.6.2 Simulation results

We shall now present simulations results in order to verify the analytical result obtained in Proposition 4.6.1 which states that, for heterogeneous sessions the rate-dependent congestion signals need to be generated at an intensity greater than a certain number. The simulations were performed using *ns-2*(version 2.26) [MF]. In the simulation scenario, nine Scalable TCP (with SACK) sessions shared a link of 200Mbps. Sessions 1,2 and 3 had a RTT of 50ms, sessions 4,5 and 6 had a RTT of 90ms, and sessions 7,8 and 9 had a RTT of 140ms. In Figure 4.6, the window size is plotted as a function of time for different values of q, the packet drop probability. We note that $q = 0$ corresponds to losses which occur when the capacity is achieved. It was observed that these losses were not always synchronous. However, during periods of synchronous losses (which have been pointed out in the figure) the rate ratio of the sessions with the largest RTT remains constant. Even though there are asynchronous losses due to congestion, the window sizes of the sessions with larger RTTs go towards zero. We must mention that in practice the TCP has a minimum sending window which is not zero. We now induce further asynchronous losses by dropping each packet with probability $q > 0$. In Figures 4.6(b)–4.6(c) there is a marked improvement in the throughput obtained by sessions with larger RTTs as the loss probability is increased. For sessions with RTT of 50ms, we observe that the periods with periods with synchronous losses are not pronounced as was the case when $q = 0$. For a larger loss probability (i.e., $q = 0.0003$), the average window size of

each sessions is almost the same. This confirms the analytical result which stated that the fairness in systems with heterogeneous MIMD sessions the fairness index can be improved by introducing asynchronous congestion signals at a sufficiently large intensity. Let γ_1, γ_2 and γ_3 be the total

Table 4.3: Throughput for each RTT class and overall efficiency

q	γ_1(Mbps)	γ_2(Mbps)	γ_3(Mbps)	$\frac{\gamma_1+\gamma_2+\gamma_3}{C}$
0	178	2.8	1	0.91
0.00015	148	25.5	7.14	0.905
0.0003	101	48	28.4	0.89

throughput of sessions with RTT of $50ms$, 90ms and 140ms, respectively. In Table 4.3, the values of throughput and the overall efficiency are given. We note that, for $q = 0.0003$, the ratio of the throughputs of two different classes, $\frac{\gamma_i}{\gamma_j}$, are almost in proportion of the respective RTTs.

Remark 4.6.1 (Window-dependent versus rate-dependent) *In the simulations we observed that the average window sizes of different sessions were almost equal when packets were dropped with probability q. When packets are dropped with a fixed probability, the losses induced are window-dependent and not rate-dependent. Hence, the effect of the asynchronous congestion losses was to improve the window-based fairness index in such a way so as to equalize the window sizes. In order to equalize the average rates, we would need to introduce rate-dependent losses, which would require the knowledge of the rates of each session. We would like to note that the analysis that we performed using a rate-based notation holds for the window-based notation as well, i.e., we could assume $\mathsf{s}(t)$ to be the window ratio process instead of the rate ratio process and the probability of the session receiving a congestion signal to be proportional to its window size.*

4.7 Inter-algorithm fairness of the MIMD algorithm: equal RTTs

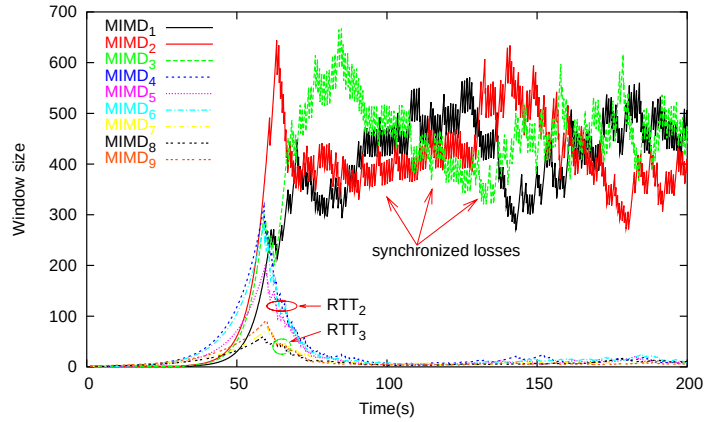
In the second part of this chapter, we shall study the bandwidth sharing when sessions using the MIMD algorithm and sessions using the AIMD algorithm share the capacity link, and the losses are *synchronous*. Specifically, we study the equilibrium behaviour of the window size, and the throughput obtained by a session of each algorithm at equilibrium in the presence of synchronous losses only. We also provide conditions under which a source of one algorithm can obtain a better throughput than a source of the other algorithm. In this section, we shall assume that each session has the same RTT, τ.

Remark 4.7.1 *We note that we shall be using a window-based notation in the rest of this chapter. For sessions with equal RTTs, the window-based notation is equivalent to rate-based notation.*

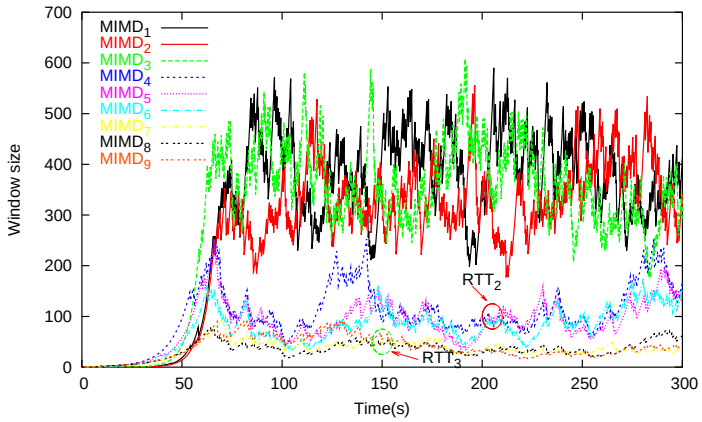
4.7.1 System model

Let us consider l sessions which share a link of capacity C *bits/s*. Each session transmits data using packets of size M bits. Let Λ be the bandwidth-delay product (BDP) of the system. We shall assume that each session has the same RTT, τ, and that the RTT is mainly determined by the propagation delay and, hence, can be considered to be a constant. We have the relation

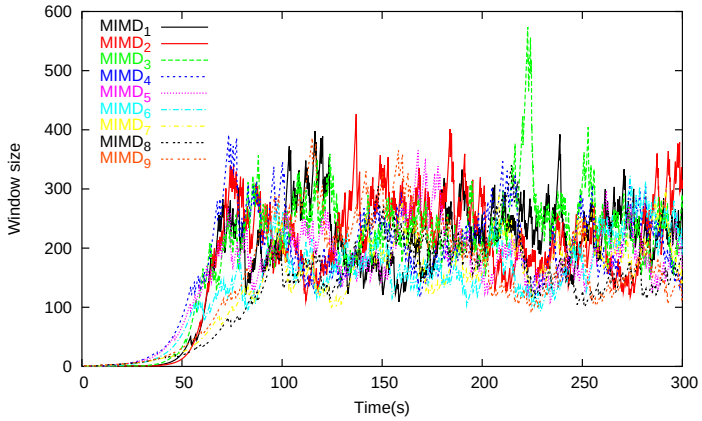
$$\Lambda = \frac{C \cdot \tau}{M}. \tag{4.88}$$



(a) $q = 0$.



(b) $q = 0.00015$.



(c) $q = 0.0003$.

Figure 4.6: Window evolution of MIMD sessions. $\mathrm{RTT}_1 = 50ms$. $\mathrm{RTT}_2 = 90ms$. $\mathrm{RTT}_3 = 140ms$.

Let $\mathsf{x}(t) = (\mathsf{x}_1(t)\ \mathsf{x}_2(t)\ \ldots\ \mathsf{x}_l(t))$ denote the vector of *window sizes* of the l sessions at time t. A synchronous loss (i.e., a loss for each session) is assumed to occur at time t if

$$\sum_{i=1}^{l} \mathsf{x}_i(t) \geq \Lambda. \tag{4.89}$$

The above condition is equivalent to saying that a synchronous loss occurs when the total number of outstanding packets in the network exceeds the total number of packets that the system can handle.

Without loss of generality, let sessions $1, 2, ..., k$ use the MIMD congestion control algorithm and the rest $l - k$ sessions use the AIMD congestion control algorithm. In the absence of losses, the two algorithms increase the window in the following way

$$\mathsf{x}_i(t+\Delta) = \begin{cases} \mathsf{x}(t)\alpha_m^{\Delta/\tau} & \text{for } 1 \leq i \leq k \\ \mathsf{x}(t) + \alpha_a \frac{\Delta}{\tau}, & \text{for } k+1 \leq i \leq l, \end{cases} \tag{4.90}$$

where α_m and α_a are the increase parameters of the MIMD and the AIMD algorithm, respectively. For example, $\alpha_m = 1.01$ for Scalable TCP, and $\alpha_a = 1$ for standard TCP. Let t_n denote the time instant when the nth congestion signal is received. We note that a congestion signal is equivalent to a synchronous loss. In response to a congestion signal the two algorithms decrease the window in the following way

$$\mathsf{x}_i(t_n^+) = \begin{cases} \beta_m \mathsf{x}(t_n) & \text{for } 1 \leq i \leq k, \\ \beta_a \mathsf{x}(t_n) & \text{for } k+1 \leq i \leq l, \end{cases} \tag{4.91}$$

where β_m and β_a are the decrease parameters of the MIMD and the AIMD algorithm, respectively. For example, $\beta_m = 0.875$ for Scalable TCP, and $\beta_a = 0.5$ for standard TCP.

Let $x(n)$ denote the window size vector embedded just after the nth congestion signal is received, i.e, $x(n) = \mathsf{x}(t_n+)$. Let δ_n denote the time between the nth and $(n+1)$th congestion signals. Since all the sessions are assumed to receive congestion signals at the same instant, we can write the following recursive equation for $x(n)$.

$$x_i(n+1) = \begin{cases} \beta_m x_i(n) \alpha_m^{\delta_n/\tau} & \text{for } 1 \leq i \leq k, \\ \beta_a \left(x_i(n) + \alpha_a \frac{\delta_n}{\tau} \right) & \text{for } k+1 \leq i \leq l. \end{cases} \tag{4.92}$$

4.7.2 Bandwidth sharing

The transient behaviour of the window sizes can be obtained by solving Equation (4.92) along with Equation (4.89). Given an initial window vector $x(0)$, the time to the first loss t_1 and, hence, $x(1)$ can be computed. This way we can recursively compute $x(n)$. This allows us to obtain the behaviour of the window size vector and the loss instants before the equilibrium is reached. At equilibrium, δ_n and $x(n)$ will converge to their steady state values denoted by δ^* and ψ, respectively. We are interested in finding the window size, ψ_i, of each session at equilibrium. Then, ψ_i together with δ^* will allow us to obtain the throughput for session i. At equilibrium $x(n)$ would be identical to $x(n+1)$, $x(n+2)$, and so on. Therefore, for each session i, we can obtain ψ_i from Equation (4.92) as follows:

$$\psi_i = \begin{cases} \beta_m \psi_i \alpha_m^{\delta^*/\tau} & \text{for } 1 \leq i \leq k, \\ \beta_a(\psi_i + \alpha_a \delta^*/\tau) & \text{for } k+1 \leq i \leq l. \end{cases} \tag{4.93}$$

The l equations in Equation (4.93) are fixed point solutions of the corresponding equations in Equation (4.92). We now have $l+1$ variables, $\psi_i, 1 \leq i \leq l$ and δ^*, and l equations. The final equation can be obtained by noting that a loss occurs when the condition in Equation (4.89) is satisfied. Therefore, the $(l+1)$th equation is obtained as

$$\sum_{i=1}^{k} \frac{\psi_i}{\beta_m} + \sum_{i=k+1}^{l} \frac{\psi_i}{\beta_a} = \Lambda. \tag{4.94}$$

We note that from any one of the first k equations in (4.93) we can obtain the value of δ^*. However, the variables $\psi_i, 1 \leq i \leq k$ cannot be uniquely determined from these k equations. They will depend on the window vector just after the first synchronous loss. This result is equivalent to the result obtained in [CJ89] where the rate vector of homogeneous MIMD sessions was dependent on the initial rate vector. However, the equilibrium window size for the AIMD sessions can be uniquely determined from Equation (4.93), using which we obtain

$$\delta^* = \tau \frac{\log[1/\beta_m]}{\log \alpha_m}, \tag{4.95}$$

$$\psi_i = \alpha_a \frac{\beta_a}{1-\beta_a} \frac{\log[1/\beta_m]}{\log \alpha_m} \quad k+1 \leq i \leq l. \tag{4.96}$$

Nevertheless, from (4.94) we can obtain the sum of the equilibrium window sizes of the MIMD sessions as

$$\sum_{i=1}^{k} \psi_i = \beta_m \Lambda - \frac{\beta_m}{\beta_a} \sum_{j=k+1}^{l} \psi_j, \tag{4.97}$$

In order to compute the throughput, γ_i, of session i, we divide the time interval δ^* in slots of length τ. We note that, just after a loss instant, the window size for session i is ψ_i. In between two loss instants, the window size for each session increases according to the algorithm given in Equation (4.90). Also, in every RTT (i.e., in every slot), session i transfers packets equivalent to its present window size. Therefore, in between two loss instants, the total number of packets that are transferred by session i can be obtained by summing the window sizes during the δ^*/τ RTTs. As before, we can obtain the throughput γ_i for each AIMD session, whereas for the MIMD sessions we can only obtain the total throughput, γ_m.

$$\gamma_m = \frac{M}{\delta^*} \sum_{i=1}^{k} \psi_i \frac{\alpha_m^{\lfloor \frac{\delta^*}{\tau} \rfloor + 1} - 1}{\alpha_m - 1} \tag{4.98}$$

$$\gamma_i = \frac{M \alpha_a}{\tau} \left(\psi_i + \alpha_a \frac{\lfloor \frac{\delta^*}{\tau} \rfloor + 1}{2} \right) \quad k+1 \leq i \leq l. \tag{4.99}$$

We note that the throughput expressions are approximate since the number of packets transferred in an RTT is an integer whereas ψ_i can take non-integer values. Also, the number of packets transferred in the RTT in which a loss occurs may not be equal to $\lfloor \psi_i \rfloor$.

We can make the following observations from Equation (4.95)-Equation (4.99):

1. The equilibrium value of the time between two loss instants, δ^*, is independent of the parameters of the AIMD algorithm. It is determined by the RTT, τ, and the parameters of the MIMD algorithm only.

Table 4.4: Several MIMD and several AIMD sessions.

	Link Speed (Mbps)	δ (s)	γ_m (Mbps)	γ_a (Mbps)	ψ_a (packets)
$l = 6.\ k = 3.$	200	2.83 (3)	164(151)	3.5(3.4)	29 (27)
$l = 12.\ k = 6.$	300	2.83 (3)	238.6(218)	3.5(3.4)	29 (27)

2. The equilibrium window size of the AIMD sessions depends only on the increase and decrease parameters of the two algorithms. It does not depend on the capacity. Also, the AIMD sessions have the same equilibrium window behaviour. Hence, the AIMD sessions obtain identical throughputs.

3. The MIMD session utilise the rest of the capacity.

4.7.3 Simulation results

We now compare these observations with simulations performed using *ns-2* (version 2.26). Unless stated otherwise, in the simulations we used the same set of parameters. The MIMD sessions used Scalable TCP, and the AIMD sessions used TCP NewReno. The packet size, M, for each session was set to 1040 bytes (1000 bytes of data + 40 bytes of header). The propagation delay[13], τ, was taken to be $100ms$. The increase and decrease parameters for the two algorithms were set to $\alpha_m = 1.01$, $\alpha_a = 1.0$, $\beta_m = 0.75$, and $\beta_a = 0.5$. Since the ψ_i for AIMD increases with decrease in β_m, we set β_m to a value smaller than its recommended value so that the AIMD sessions also obtain a certain throughput. From Figure 4.7(a) (3 MIMD sessions and 3 AIMD sessions) and Figure 4.7(b) (6 MIMD sessions and 6 AIMD sessions), we note that the window size of AIMD sessions sampled just after loss instants indeed converges to the same equilibrium value whereas the equilibrium window size of an MIMD session depends on its window size just before the first synchronous loss. The ψ_i for AIMD sessions remains the same even though the link capacity is increased from 200Mbps to 300Mbps and the total number of sessions is increased from six to twelve. Let γ_a and ψ_a denote the throughput and the equilibrium window size of any one of the AIMD sessions, respectively.

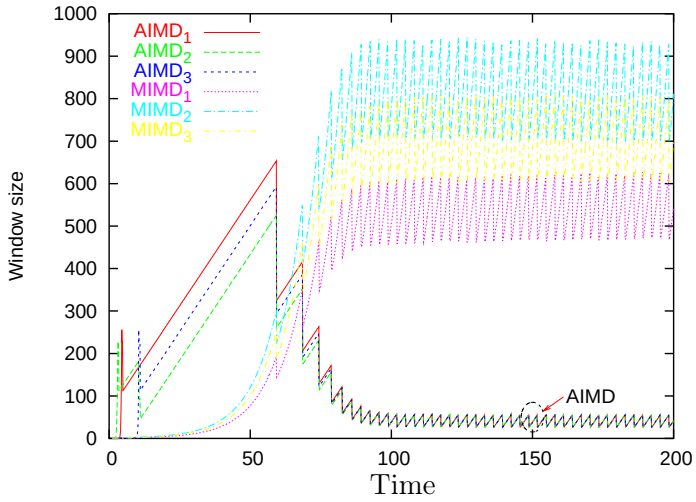
In Table 4.4, the analytical and simulation values of δ^*, γ_m, γ_a, and ψ_a are given. The simulation values are given in parentheses. As predicted in the analysis, the equilibrium window size and the throughput of the AIMD sessions remains unchanged even when the capacity is increase from 200Mbps to 300Mbps, and the total number of sessions is increased from six to twelve. We note that γ_m represents the total throughput of the MIMD sessions.

4.7.4 Throughput comparison

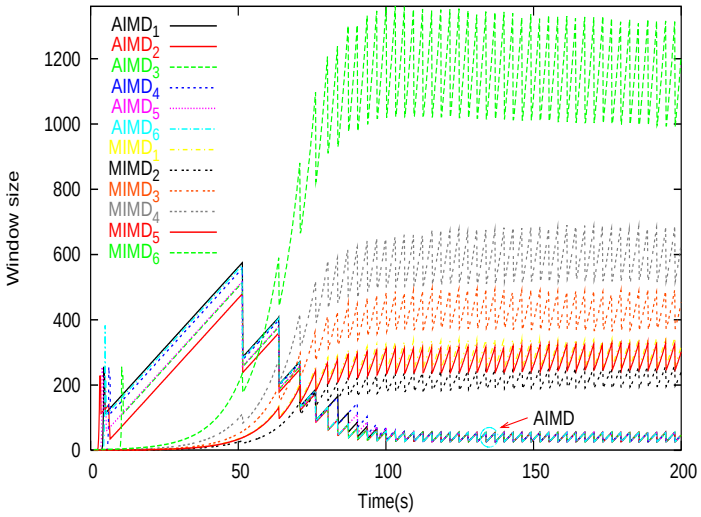
We now study the scenario where one MIMD source and AIMD source share the same link. We note that each source can initiate several sessions of the same algorithm. We are interested in knowing the conditions under which the AIMD source can obtain higher throughput than the MIMD source.

First, we consider the case in which each source initiates only one session. In such a scenario, the window size and the throughput of each session is obtained from Equations (4.96–4.98) with $l = 2$ and $k = 1$.

[13]We have assumed that the propagation delay is the same as the RTT.



(a) $C = 200$Mbps. $l = 6$. $k = 3$.



(b) $C = 300$Mbps. $l = 12$. $k = 6$.

Figure 4.7: Window evolution of sessions.

From Equations (4.98–4.99), as $\Lambda \to \infty$ (i.e., $C \to \infty$), the ratio of the throughputs, γ_2/γ_1, goes to 0. This suggests that in high capacity systems, the MIMD source will get most of the capacity. On the other hand, if the BDP of the system is small, the MIMD source will obtain a lower throughput compared to the AIMD session.

Proposition 4.7.1 *Let Λ_l denote a threshold BDP below which an AIMD session will get higher throughput compared to an MIMD session. The threshold value, Λ_l, is given by*

$$\Lambda_l = \psi_2 \left(\frac{\alpha_a \delta^*}{2\tau} \left(\frac{\delta^*}{\tau} + 1 \right) \left(\frac{\alpha_m - 1}{\alpha_m - \beta_m} \right) + \frac{1}{\beta_a} \right). \tag{4.100}$$

Proof. The above relation can be obtained using the fact $\gamma_1 \leq \gamma_2$ together with Equation (4.97). ■

The value of Λ_l depends only on the increase and decrease parameters of the two algorithms. In Table 4.5, we give the values of Λ_l for different β_m with $\alpha_m = 1.01$ and $\alpha_a = 1$. In Figure 4.8(a),

Table 4.5: Λ_l for different values of β_m.

β_m	0.875	0.75	0.5
Λ_l	47.34	106.6	282.73

the window evolution of the two sessions is shown for $C = 13$Mbps and $\beta_m = 0.5$. For this system, the BDP is less than the Λ_l. The AIMD algorithm obtains a better throughput in this case. In the next set of simulations, we set β_m to its recommended value of 0.875. In Figure 4.8(b), the corresponding window evolution is shown. The effect of increasing β_m is to reduce the share of the AIMD session. In Table 4.6, a comparison of the values obtained from analysis and simulations is presented.

From Equation (4.99), it was inferred that the throughput obtained by each AIMD session remains constant whereas the total throughput of the MIMD sessions increases with increase in capacity. An AIMD source may want to obtain throughputs similar to an MIMD source. In this case, the AIMD source may open several sessions in order to improve its observed throughput. Since each AIMD session gets the same throughput independent of the number of AIMD sessions (assuming there is sufficient capacity), an AIMD source can improve its observed throughput by opening multiple sessions.

Proposition 4.7.2 *The smallest number of sessions, ν, with which an AIMD source will obtain a better global throughput compared to single MIMD source is given by*

$$\nu = \left\lceil \frac{\beta_m(\alpha_m - \beta_m)}{\psi_a \alpha_a \lfloor \frac{\delta^*}{\tau} \rfloor (\lfloor \frac{\delta^*}{\tau} \rfloor + 1)(\alpha_m - 1} \left(\Lambda - \frac{\psi_a}{\beta_a} \right) \right\rceil, \tag{4.101}$$

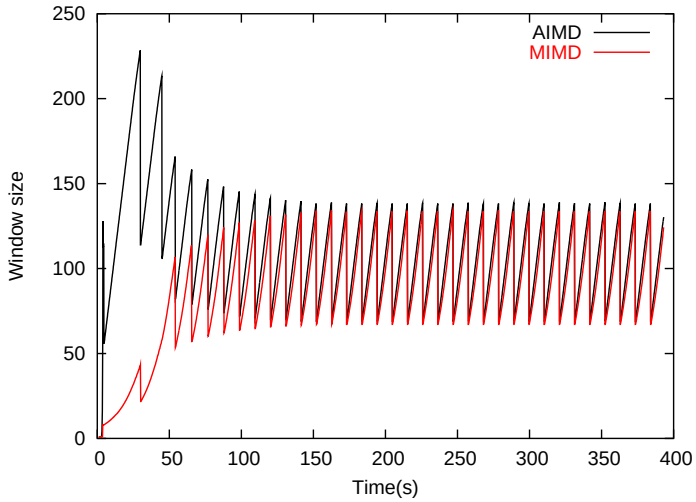
where ψ_a is the equilibrium window of any one of the AIMD sessions and is defined in Equation (4.96).

We note that ν depends only on Λ and the increase and decrease parameters of the two algorithms. In Table 4.7 we give the value of ν for different values of Λ for $\beta_m = 0.875$.

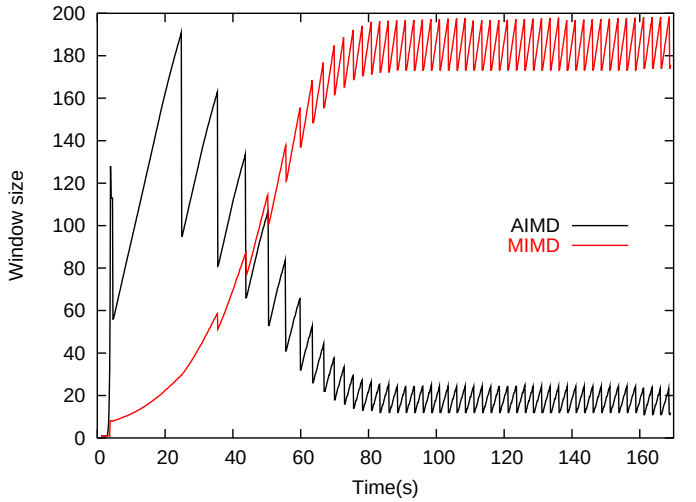
Similar to the AIMD source, an MIMD source may also try to improve its observed throughput by opening several sessions. Since, from Equation (4.99), the AIMD source will get a throughput independent of the number of MIMD sessions, the observed throughput of an MIMD source will

Table 4.6: One MIMD and one AIMD session. $\tau = 140$ms.

	Link Speed (Mbps)	η_1 (Mbps)	η_2 (Mbps)	ψ_1 (packets)	ψ_2 (packets)
$\beta = 0.875$	3	2.22 (2.36)	0.6 (0.52)	66.2 (70.7)	13.41 (11)
	5	4.05 (4.1)	0.75 (0.71)	96.82 (98)	13.41 (12)
	10	8.86 (8.74)	0.92 (0.86)	173.39 (173)	13.41 (12)
$\beta = 0.5$	13	4.73 (5.19)	4.92 (5.64)	69.09 (67)	69.6 (69)
	15	6.08 (6.62)	5.04 (5.65)	86.59 (84)	69.6 (68)
	30	16.64 (16.95)	5.47 (5.86)	217.83 (211)	69.6 (69)



(a) $C = 13$Mbps. $\beta_m = 0.5$. $\tau = 140$ms.



(b) $C = 10$Mbps. $\beta_m = 0.875$. $\tau = 140$ms.

Figure 4.8: Window evolution for one MIMD session and one AIMD session.

not improve by opening several sessions. This result is in contrast to the result obtained in Equation (4.101), where we noted that an AIMD source can improve its observed throughput by opening several sessions.

4.8 Inter-algorithm fairness of the MIMD algorithm: unequal RTTs

In this section we study the effect of having a different RTT for each session on the equilibrium window behaviour. The notation used and the scenario is the same as in Section 4.7. We shall assume that there exists a BDP, Λ, such that there is a synchronous loss when condition in Equation (4.89) is satisfied. Let τ_i be the RTT of session i. Then, we can rewrite Equation (4.93) as

Table 4.7: ν for different values of Λ.

$\beta_m =$	Λ	100	500	1000	10000	50000
0.875	ν	3	18	37	372	1863
$\beta_m =$	Λ	100	500	1000	10000	50000
0.5	ν	1	3	7	71	358

follows:

$$\psi_i = \beta_m \psi_i \alpha_m^{\delta^*/\tau_i},\ 1 \le i \le k, \tag{4.102}$$

$$\psi_i = \beta_a(\psi_i + \alpha_a \delta^*/\tau_i),\ k+1 \le i \le l. \tag{4.103}$$

The expressions for throughput are:

$$\gamma_i = \begin{cases} \frac{M}{\delta^*}\psi_i \sum_{j=0}^{\lfloor \frac{\delta^*}{\tau_i} \rfloor} \alpha_m^j & \text{for } 1 \le i \le k, \\ \frac{M}{\delta^*}\psi_i \alpha_a \sum_{j=0}^{\lfloor \frac{\delta^*}{\tau_i} \rfloor} j & \text{for } k+1 \le i \le l. \end{cases} \tag{4.104}$$

For Equation (4.102) to be consistent we need

$$\delta^* = \frac{\log[1/\beta_m]}{\log[\alpha_m]} \min_{1 \le i \le k} \tau_i. \tag{4.105}$$

Therefore, among the MIMD sessions, only the session with the least RTT will have an equilibrium window size different from zero. The equilibrium window size of the other MIMD sessions will go to zero. Hence, we can consider the case where there is only one MIMD session and several AIMD sessions.

For $k = 1$, from Equations (4.102–4.103), we obtain

$$\begin{aligned} \delta^* &= \tau_1 \frac{\log[1/\beta_m]}{\log \alpha_m}, \\ \psi_i &= \alpha_a \frac{\beta_a}{1-\beta_a} \frac{\log[1/\beta_a]}{\log \alpha_a} \frac{\tau_1}{\tau_i}, \quad 2 \le i \le l, \\ \psi_1 &= \beta_m \Lambda - \frac{\beta_m}{\beta_a} \sum_{i=2}^{l} \psi_i. \end{aligned}$$

The inter-loss time depends entirely on the parameters and the RTT of the MIMD session. The effect of different RTTs for the AIMD sessions is to scale ψ_i by a factor of τ_1/τ_i. Therefore, an AIMD session with lower RTT can obtain a better throughput.

4.8.1 Several MIMD sessions

We observed in the previous subsection that if there are l MIMD sessions with different RTTs sharing a link, then the session with the smallest RTT will get all the capacity and the window sizes for other sessions will go to zero. This result was also mentioned in [XHR04]. However, if the sessions have the liberty to choose their increase and decrease parameters then each session can obtain some share of the capacity. Let α_{mi} and β_{mi} be the increase parameter and the decrease parameter, respectively, of the ith MIMD session.

Proposition 4.8.1 *A system with l MIMD sessions having different RTTs will have a behaviour similar to a system with l MIMD sessions having the same RTT if*

$$\tau_i \frac{\log[1/\beta_{mi}]}{\log[\alpha_{mi}]} = \textit{(a constant)}. \tag{4.106}$$

The inter-loss time, δ^*, will then be equal to this constant.

Proof. We note that with this value of δ^*, Equation (4.102) is consistent for non-zero $\psi_i,\ 1 \leq i \leq l$. Therefore, an equilibrium solution exists. Let $x(0)$ is the initial window size vector. The time to the first synchronous loss, t_1, can be computed using the condition $\sum_{i=0}^{l} x_i(0)\alpha_{mi}^{t_1/\tau_i} = \Lambda$. We can now compute $x_i(1) = \beta_{mi}x_i(0)\alpha_{mi}^{t_1/\tau_i}$. The next loss will occur after a time δ^*. This can be verified by noting that $\sum_{i=0}^{l} \frac{x_i(1)}{\beta_{mi}} = \Lambda$, and $t_2 = \delta^*$ given by Equation (4.106) satisfies $\sum_{i=0}^{l} x_i(1)\alpha_m^{t_2/\tau} = \Lambda$. From this we obtain $t_2 = \delta^*$. Since t_1 is the equilibrium value of the inter-loss time, $x_i(1)$ will also be the equilibrium value of ψ_i. Now, the system will be similar to the same RTT case where the equilibrium window size vector is the same as the window size vector just after the first synchronous loss. ■

Therefore we can obtain a condition on setting the increase and decrease parameters of MIMD algorithm as a function the RTT in order to obtain the same performance as system with the same RTT.

4.9 Conclusions

In the first part of the chapter, we studied the fairness properties of systems with only MIMD sessions. For homogeneous sessions, it was observed that the fairness index of a system with synchronous congestion signals could be improved by introducing rate-dependent losses with non-zero intensity. For heterogeneous sessions, i.e., sessions with different RTTs, it was observed that the intensity of the rate-dependent congestion signals needed to be greater than a threshold intensity in order to improve the fairness index. In the second part of this chapter, we studied bandwidth sharing between MIMD sessions and AIMD sessions in systems with synchronous losses only. It was noted that for a given set of parameters, AIMD sessions obtained a throughput which is independent of the BDP, and that the rest of the capacity was utilized by the MIMD sessions. In networks with BDP less than a threshold value, it was observed that one AIMD session obtained better throughput than one MIMD session. It was also observed that an AIMD source could open multiple sessions in order to improve its observed throughput whereas for the MIMD source the throughput was invariant to the number of sessions it opened.

Chapter 5

A singular perturbation based analysis of a RED queue

Several Active Queue Management (AQM) techniques for routers in the Internet have been proposed and studied during the past few years. One of the widely studied proposals, Random Early Detection (RED), involves dropping an incoming packet with some probability based on the estimated average queue length at the router. The analytical approaches to obtaining average drop probabilities in a RED enabled queue have been either based on using the instantaneous queue size for calculating the drop probability or have considered averaging with a fluid approximation. In this chapter, we use a singular perturbation based approach to analyse a RED enabled queue with drop probabilities based on the estimated average queue size as has been proposed in the standard RED. The singular perturbation approach is motivated by the fact that the instantaneous and the estimated average queue lengths evolve at two different time scales. We present an analytical method to calculate the average queue length and the average drop probability for the non responsive flows. We also provide analytical expressions when the arrival are Poissonian and service times are exponential. Our model is derived under several approximations, and is validated through simulations.

The results in this chapter have appeared in [AAP04].

5.1 Introduction

In the previous chapters, we studied the performance measures for an end-user of the Internet while assuming the network to be a black box which gave feedback signals. In particular, we studied the window size behaviour of congestion control algorithms which received congestion signals from the network. A network consists of a set of links and nodes, called routers, which receive packets on incoming links and forward the packets on one of the outgoing links. In order to counter the fluctuations in the rate of incoming packets, a router is provisioned with a buffer which can absorb a burst of packets of limited number. The overflow of this buffer leads to congestion signals being generated. In this chapter, we invert the scenario considered in previous chapters. Specifically, we consider the behaviour of the queue length in a router[1] while modelling the end-users as one

[1] An element, which has several outgoing links, that switches incoming packets to one of the outgoing links

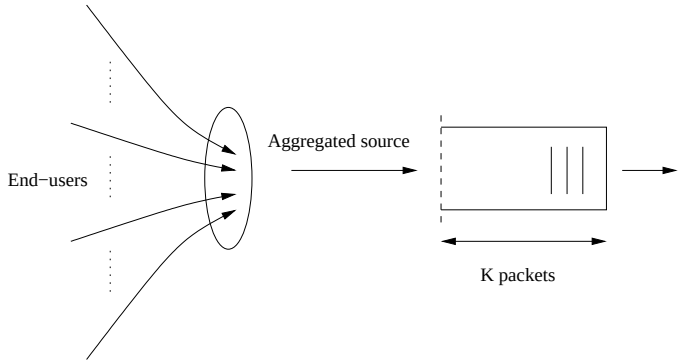


Figure 5.1: The incoming traffic as perceived by a router.

aggregate source which sends packets to this router. An illustration of this model is shown in Figure 5.1.

Since a router only has a finite buffer, it has to exercise some admission control, on the incoming packets. We note that an admission control policy (or, a queue management scheme) is, in effect, a policy to generate congestion signals. Thus, the choice of the queue management scheme affects the behaviour of the window size of a congestion control algorithm. A simple and easy to implement queue management scheme, called the drop-tail scheme, admits packets as long as there is space available in the buffer. This scheme was also implemented in routers in the Internet. However, it was observed that TCP used in conjunction with drop tail queues in the routers led to oscillatory behaviour of the queue length and to bursty packet losses at the routers [ZC90]. A burst of packet losses meant that several sources would increase and decrease their window size in unison. The synchronized increase and decrease of the window size by different sources led to inefficient use of the outgoing link. Thus, Active Queue Management (AQM) schemes were proposed to improve the link utilization by sending congestion signals in anticipation of congestion (see, for example, [FJ93], [PPP00], and [FKSS02]).

In order to overcome the synchronised window evolution, the Random Early Detection (RED) algorithm proposes to drop packets at random before congestion actually sets in [FJ93]. Two of the design aims of RED are to accept occasional bursts and to maintain a reasonable average queue length when the system is heavily loaded with the objective of efficient use of the system capacity. Towards this end, the routers maintain a variable corresponding to the exponentially weighted moving average of the instantaneous queue length. An incoming packet is dropped with some probability which is a function of this estimated average queue length. The performance of the RED algorithm depends on the setting of parameters and the traffic characteristics [CJOS01]. Several alternatives and modifications have been proposed in [FKSS02], [ALLY01], [TM04], and [OLW99].

The aim of our study is to use singular perturbation techniques (see, for example, [Sch86] and [AFH02]) to provide an analytical expression for computing the drop probabilities and average queue length in a RED enabled queue with averaging as has been proposed in RED. Furthermore, we make use of the transition probability matrices in order to preserve the stochastic nature of the system as opposed to a fluid approximation which is deterministic. The dynamics of the average and the instantaneous queue lengths are modelled as a two dimensional Markov chain, which is then approximated by a non-homogeneous (or level dependent) Quasi-Birth-and-Death (QBD) process

[LR99]. This model then permits us to obtain the joint steady state distribution of the average and the instantaneous queue lengths. The use of singular perturbation technique is motivated by the fact that two different time scales are involved in the evolution of the average and the instantaneous queue length. Indeed, for very small values of the averaging parameter, the average queue length can be assumed to vary much more slowly compared to the instantaneous queue length. The use of this decomposition allows us to compute the steady state distribution and the desired performance metrics at a reduced complexity.

Remark 5.1.1 *We must emphasize that the model which we shall study does not take into account the effect of packet drops on the incoming rate of packets. As was mentioned earlier, congestion control algorithms react to packet drops by reducing their sending rate. Thus, the rate of incoming packets to the router is also reduced. However, there are traffic sources such as real-time streaming flows that transfer data without using a congestion control algorithm, i.e., they do not react to packet drops. Thus, our source model is more representative of sources that do not adapt their sending rates based on congestion signals.*

Next, we mention the techniques which have been previously used to analyse RED, and mention the differences with our work.

5.1.1 Related literature

Various authors have studied the behaviour of RED, both through analysis and through simulations. In [BMB00], the authors analyse the performance of a RED queue using the instantaneous queue length, instead of the average queue length, to compute the drop probabilities. In [MGT00] and [HMTG01], a fluid approximation is used to study the effect of the exponential averaging. In [KRV02], the authors compute the joint distribution of the instantaneous and the averaged queue length of an $M/M/1/K$ queue as a solution of a set of differential equations. An analytical expression for the case of buffer size 1 and 2 is also provided. However, they consider packet drops only due to buffer overflow. The study of [SVL02] uses a time scale decomposition to obtain in the limit a differential equation for the exponentially averaged queue length. They also provide a diffusion approximation to quantify the error due to the ODE approximation. This method of analysis is then extended in [LV00] in which the assumption of a small averaging parameter is relaxed and a time dependent arrival rate is considered. In [KLVK01], the authors extend this model to consider the aggregate sending rate of TCP sources which react to packet drops.

Our method is also based on noting that average and the instantaneous queue lengths evolve at different time scales. However, unlike the ODE approximation obtained in [SVL02], we use a discretized version of that fluid dynamics which has a simple probabilistic interpretation. By introducing probabilistic dynamics, we model the joint average and instantaneous queue length process as a non-homogeneous QBD process. This formulation then allows us to obtain the joint distribution of the average and the instantaneous queue length process. We note that this type of discretization is very frequent in numerical solutions of fluid models and of diffusions, including controlled ones and there is a huge literature that provides conditions under which the stochastic processes related to the discrete model converge (in various senses) to the original fluid model as the discretization step goes to zero, see, e.g., Theorems 2.7 and 4.2 in [CF96] (or the Appendix of [BCD97] as well as [KD92]).

The rest of the chapter is organized as follows. In Section 5.2 we describe the RED algorithm and present the system model. In Section 5.3 we outline the analytical method to compute the joint distribution of the average and instantaneous queue length, and to compute other performance

metrics. In Section 5.4 we compare the results obtained using the analysis and with simulations. Finally, in Section 5.5 we present the conclusions.

5.2 Problem formulation

We first describe the algorithm which a RED-enabled router uses to drop packets.

5.2.1 The RED algorithm

The basic algorithm of RED was proposed in [FJ93] and can be summarized as follows. The router updates an exponentially averaged queue length variable, q_n, where the subscript n denotes the nth packet arrival. In addition to q_n, the router also keeps a count, c_n, of the number of packets accepted since the last dropped packet. The incoming packet is dropped with a probability, p_d, which is computed according to the following algorithm.

```
if q_n < min_th then
    p_d = 0, c_n = -1,
else if min_th ≤ q_n ≤ max_th then
    p_b = max_p(q_n − min_th)/(max_th − min_th),
    p_d = min(1, p_b/(1 − c_n p_b)),
else
    p_d = 1,
```

where max_p (maximum drop probability), min_{th} (minimum threshold for queue), and max_{th} (maximum threshold for queue) are given constants. The variable c_n is set to zero each time an incoming packet is dropped. The estimated average queue length, q_n, is an exponentially weighted moving average of the instantaneous queue length, Q_n, and is computed as follows.

```
if Q_n == 0 then
    q_{n+1} = (1 − α)^w q_n,
else
```

$$q_{n+1} = (1-\alpha)q_n + \alpha Q_{n+1}, \tag{5.1}$$

where α is the queue weight, and w is the average number of packets the router could have transmitted during the previous idle period. Figure 5.2 shows the drop function associated with the basic RED algorithm.

5.2.2 The system model

Let us consider a RED-enabled router which has a buffer size of K packets. We note that the maximum queue size here is in terms of packets and not bits. Let T_n denote the inter-arrival time between the $(n-1)$th and nth arrival. The sequence $\{T_n, n \geq 1\}$ is assumed to be independent and identically distributed with mean λ^{-1}, distribution function $T(x)$, and Laplace-Stieltjes transform $T^*(s)$. The packet service times are assumed to be exponentially distributed with mean μ^{-1}. In the notation of queueing theory, the model corresponds to a $G/M/1/K$ queue with RED queue management scheme.

Assumption 5.2.1 *For $min_{th} \leq q_n \leq max_{th}$, an incoming packet is dropped with a probability p_d equal to p_b.*

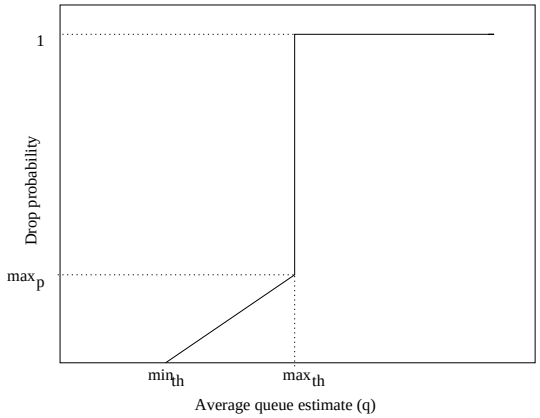


Figure 5.2: RED drop probability function

Remark 5.2.1 *In [FJ93], the authors compare two methods of calculating the drop probability, p_d. In the first method, p_d is taken to be equal to p_b, and in the second method p_d is computed as in the algorithm described in Section 5.2.1. They conclude that, for the same number of average dropped packet, the losses are more spread out when the second method is used. Therefore, by assuming $p_d = p_b$ we can expect to obtain an upper bound on the probability of more than j successive packet losses.*

Let Q_n and q_n be the instantaneous and estimated average queue lengths, respectively, just before the nth arrival. The recursive equations for the evolution of the state (q_n, Q_n) are given by

$$Q_{n+1} = \begin{cases} \max(Q_n - D_n, 0) & \text{w.p.} \quad h(q_n) \quad \text{or if } Q_n = K; \\ \max(Q_n + 1 - D_n, 0) & \text{w.p.} \quad 1 - h(q_n), \end{cases} \tag{5.2}$$

$$q_{n+1} = (1-\alpha)q_n + \alpha Q_{n+1}, \tag{5.3}$$

where D_n is the number of packet departures during the inter-arrival time T_{n+1}, and $h(\cdot)$ denotes the RED packet drop probability function, i.e., $p_d = h(q_n)$.

With this formulation, the process $\{(q_n, Q_n), n \geq 0\}$ is a Markov chain. Its analysis is difficult, partly because the support of q is a countable subset of the interval $[0, K]$ where K is the maximum queue size.

We propose an approximation that reduces the support of q to a finite set denoted by $V_m = \{0, ..., mK\}$, where m is some integer. In the following discussion, $\hat{q}_n$ will refer to the discretized estimated average queue length. We assume below that α is sufficiently small such that $\alpha mK < 1$. The recursive equation for $\hat{q}_n$ is given by

$$\hat{q}_{n+1} = (1-\alpha)\hat{q}_n + \alpha m Q_{n+1}, \quad \hat{q}_n \in \{0, 1, ..., mK\}. \tag{5.4}$$

Now, we replace Equation (5.4) by the approximation

$$\hat{q}_{n+1} = \hat{q}_n + C_n, \tag{5.5}$$

where

$$C_n = \begin{cases} 1 & \text{w.p. } \alpha(mQ_{n+1} - \hat{q}_n)^+ \ ; \\ -1 & \text{w.p. } \alpha(\hat{q}_n - mQ_{n+1})^+ \ ; \\ 0 & \text{otherwise} \ , \end{cases}$$

with $x^+ = \max(x, 0)$.

We note that with this definition, we obtain

$$\begin{aligned} E[\hat{q}_{n+1} - \hat{q}_n | \hat{q}_n, Q_{n+1}] &= 1 \cdot \alpha(mQ_{n+1} - \hat{q}_n)^+ + (-1) \cdot \alpha(\hat{q}_n - mQ_{n+1})^+ \\ &= \alpha(mQ_{n+1} - \hat{q}_n), \end{aligned} \tag{5.6}$$

so that Equation (5.4) becomes the mean field of Equation (5.5). With this approximation the transitions of $\hat{q}_n$ conditioned on Q_n become stochastic as opposed to the deterministic transitions in the original system.

The motivation for such an approximation is that we are interested in obtaining the steady state distribution of the couple $(\hat{q}_n, Q_n)$ which will allow us to study the various performance measures (e.g., average queue length, average drop probability, etc.) of the system. With this approximation we can model the $(\hat{q}_n, Q_n)$ process as a non-homogeneous QBD process for which algorithms are available to efficiently compute the steady state distribution [GJL84],[LR99].

In order to model the process $(\hat{q}_n, Q_n)$ as a QBD process, we first obtain the transition matrix of Q_n for a given value of $\hat{q}_n$. That is,

$$A_i = \{a_{jk}\}_i = P(Q_{n+1} = k | Q_n = j, \hat{q}_n = i), \qquad \forall i \in 0, 1, ..., mK. \tag{5.7}$$

The matrix A_i is a $(K+1) \times (K+1)$ matrix. Using Equation (5.2), the probability a_{jk} can be obtained as follows.

$$a_{jk} = \begin{cases} (1 - h(i))d_0 & k = \min(j+1, K); \\ (1 - h(i))d_{j-k+1} + h(i)d_{j-k} & 0 < k \leq j; \\ (1 - h(i)) \sum_{l=\min(j+1,K)}^{\infty} d_l + h(i) \sum_{l=\min(j,K)}^{\infty} d_l & k = 0, \end{cases} \tag{5.8}$$

where

$$d_l = \int_0^\infty \exp(-\mu x) \frac{(\mu x)^l}{l!} dT(x) \tag{5.9}$$

is the probability of l departures between two arrivals.

Next, we obtain the transition matrix for $\hat{q}_n$ conditioned on given values of Q_{n+1} and $\hat{q}_n$. Let C_i^+, C_i^-, C_i^0 denote matrices given by

$$\begin{aligned} C_i^+ := \{c_{jj}\} &= p(\hat{q}_{n+1} = i+1 | Q_{n+1} = j, \hat{q}_n = i) \\ &= \alpha(m \cdot j - i)^+, \\ C_i^- := \{c_{jj}\} &= p(\hat{q}_{n+1} = i-1 | Q_{n+1} = j, \hat{q}_n = i) \\ &= \alpha(i - m \cdot j)^+, \\ C_i^0 := \{c_{jj}\} &= p(\hat{q}_{n+1} = i | Q_{n+1} = j, \hat{q}_n = i) \\ &= 1 - \alpha(m \cdot j - i)^+ - \alpha(i - m \cdot j)^+, \\ &\quad \forall i \in \{0, 1, ..., mK\} \qquad \forall j \in \{0, 1, ..., K\}. \end{aligned}$$

C_i^+i, C_i^-, and C_i^0 are $(K+1) \times (K+1)$ matrices which denote the transition probability of $\hat{q}_{n+1} = \hat{q}_n + 1$, $\hat{q}_{n+1} = \hat{q}_n - 1$, and $\hat{q}_{n+1} = \hat{q}_n$, respectively, when $Q_{n+1} = j$. We note that C_i's are diagonal matrices which satisfy the equality

$$C_i^+ + C_i^- + C_i^0 = I. \tag{5.10}$$

Let $\tilde{P}$ denote the transition probability matrix for the two dimensional Markov chain $(\hat{q}_n, Q_n)$ with $\tilde{\pi}(\alpha)$ as its stationary distribution. Using the identity

$$\begin{aligned} p(\hat{q}_{n+1}, Q_{n+1}|\hat{q}_n, Q_n) &= p(\hat{q}_{n+1}|\hat{q}_n, Q_n, Q_{n+1}) \cdot p(Q_{n+1}|\hat{q}_n, Q_n) \\ &= p(\hat{q}_{n+1}|\hat{q}_n, Q_{n+1}) \cdot p(Q_{n+1}|\hat{q}_n, Q_n), \end{aligned}$$

we can write $\tilde{P}$ as

$$\tilde{P} = \begin{bmatrix} A_0(C_0^- + C_0^0) & A_0C_0^+ & 0 & \dots & \dots \\ A_1C_1^- & A_1C_1^0 & A_1C_1^+ & 0 & \dots \\ 0 & A_2C_2^- & A_2C_2^0 & A_2C_2^+ & \dots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & A_{mK}C_{mK}^- & A_{mK}(C_{mK}^0 + C_{mK}^+) \end{bmatrix}. \tag{5.11}$$

The diagonal rows entries of $\tilde{P}$ are matrices which give the transition probability matrix of Q for a given value of $\hat{q}_n$, and $\hat{q}_{n+1} = \hat{q}_n$ whereas the off-diagonal entries give the transition probability matrix of Q for a given value of $\hat{q}_n$ and $\hat{q}_{n+1} = \hat{q}_n + 1$, or $\hat{q}_{n+1} = \hat{q}_n - 1$. Using Equation (5.10), we can rewrite $\tilde{P}$ as

$$\tilde{P} = P + PC, \tag{5.12}$$

where P is given by

$$P = \begin{bmatrix} A_0 & 0 & \dots & 0 \\ 0 & A_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & A_{mK} \end{bmatrix},$$

and C is given by

$$C = \begin{bmatrix} -C_0^+ & C_0^+ & 0 & \dots \\ C_1^- & -(C_1^- + C_1^+) & C_1^+ & \dots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & C_{mK}^- & -C_{mK}^- \end{bmatrix}.$$

We note that $\tilde{P}$ is a function of the averaging parameter α. In [Flo97], the recommended value of α is of the order of 0.002. In [FGS01], the authors recommend setting

$$\alpha = 1 - \exp(-1/\Delta), \tag{5.13}$$

where Δ is the link capacity in packets/sec. For large capacity links, this suggests that the averaging parameter is small. This assumption on α allows us to obtain the steady state probabilities of $\tilde{P}$ using a singular perturbation approach which we outline below and which is computationally less expensive than algorithms to compute the invariant vectors of matrices.

5.3 Analysis

To use the singular perturbation approach, we first note that P is a matrix containing the transition probability matrices of the instantaneous queue length for a given value of the estimated average queue length when the averaging parameter $\alpha = 0$. Thus, P has mK ergodic classes each corresponding to a particular value of $\hat{q}$. The stationary distribution of each of these classes can be computed separately. This system of decomposable Markov chains is the unperturbed system which can be analysed separately for different values of $\hat{q}$. In general, the stationary distribution (which may not be unique) of the unperturbed system (i.e., $\alpha = 0$) is not the same as the stationary distribution of the original system when $\alpha \to 0$ [Sch68]. This motivates us to use singular perturbation technique to determine the stationary distribution of our system when $\alpha \to 0$.

Later, we also show that the distributions of the instantaneous queue length and the average queue length can be computed for the general case (i.e. $\alpha \nrightarrow 0$). We note that the use of the approximation given in Equation (5.5) has resulted in a transition matrix, Equation (5.11), which is similar to that of a level dependent Quasi-Birth-and-Death (QBD) process. We can now use the algorithm, given in [GJL84], for computing the steady state probability vector for level dependent QBD.

5.3.1 The limiting case

In this subsection we obtain the steady state probability vector of $\tilde{P}$ when $\alpha \to 0$. We note that all the elements of the matrices C_i^+ and C_i^- have α as a common multiple and thus we can replace C_i^+ by αD_i^+ and C_i^- by αD_i^- where D_i^+ and D_i^- have elements independent of α. Thus we can rewrite equation Equation (5.12) as

$$\tilde{P} = P + \alpha PD. \tag{5.14}$$

This is the standard formulation for the singularly perturbed Markov chains [Sch86]. The singular perturbation approach allows us to find the stationary distribution of $\tilde{P}$, $\tilde{\pi}(\alpha)$, in the limit $\alpha \to 0$.

We present the method for finding the limiting stationary distribution of $\tilde{P}$, the details of which can be found in [Sch86], [AFH02]. Let π_i denote the stationary distribution of A_i, and let V be a $(mK+1) \times (mK+1)(K+1)$ matrix such that in the ith row the entries, corresponding to the columns $\hat{q}_n = i$, are given by the rows of π_i, and is zero elsewhere. The vector π_i is the stationary distribution vector of the unperturbed Markov chain of the instantaneous queue length when packets are dropped with a probability depending on $h(i)$. Let W be a $(mK+1)(K+1) \times (mK+1)$ matrix such that in the ith column the entries corresponding to the rows $\hat{q}_n = i$ are unity and the other entries are zero.

$$V = \begin{bmatrix} \pi_0 & \underline{0} & \dots & \underline{0} \\ \underline{0} & \pi_1 & \dots & \underline{0} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{0} & \dots & \dots & \pi_{mK} \end{bmatrix}, \qquad W = \begin{bmatrix} \underline{1}' & \underline{0} & \dots & \underline{0} \\ \underline{0} & \underline{1}' & \dots & \underline{0} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{0} & \dots & \dots & \underline{1}' \end{bmatrix},$$

where $\underline{0}$ and $\underline{1}$ are $(K+1) \times 1$ vectors of 0 and 1, respectively.

Let S denote the generator matrix of the aggregated Markov chain. When α goes to 0, the stochastic sequence $\hat{q}_n$ converges weakly to a Markov chain induced by the aggregated transition

matrix. Then S is given by

$$S = VPDW.$$

Since π_i is the stationary distribution of A_i, VP reduces to V, i.e.,

$$S = VDW. \tag{5.15}$$

Denote

$$\begin{aligned} f_i^+ &= \pi_i D_i^+ \underline{1}' && (5.16)\\ &= \sum_{j=\lceil \frac{i}{m} \rceil}^{K} \pi_i(j)(mj - i). && (5.17)\\ f_i^- &= \pi_i D_i^- \underline{1}' && (5.18)\\ &= \sum_{j=0}^{\lfloor \frac{i}{m} \rfloor} \pi_i(j)(i - mj). && (5.19)\\ & && (5.20) \end{aligned}$$

where $\pi_i(j)$ is the jth element of π_i.

Using Equation (5.16) and Equation (5.18) we can rewrite S as

$$S = \begin{bmatrix} -f_0^+ & f_0^+ & 0 & \cdots \\ f_1^- & -(f_1^- + f_1^+) & f_1^+ & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & f_{mK}^- & -f_{mK}^- \end{bmatrix}.$$

The stationary distribution of this generator matrix can be obtained by solving $\gamma S = 0$. The stationary distribution, γ, is given by

$$\gamma := \{\gamma(i)\} = \gamma(0)\frac{\prod_{j=0}^{i-1} f_j^+}{\prod_{j=1}^{i} f_j^-}, \qquad i \in \{0, 1, ..., mK\}, \tag{5.21}$$

where

$$\gamma(0) = \left(\sum_{i=0}^{mK} \frac{\prod_{j=0}^{i-1} f_j^+}{\prod_{j=1}^{i} f_j^-} \right)^{-1}.$$

Proposition 5.3.1 *Let $\tilde{\pi}(\alpha)$ be the stationary distribution of $\tilde{P}$ for a given value of α, and let π_i denote the stationary distribution of A_i. Then, the limiting stationary distribution of $\tilde{P}$, $\tilde{\pi}(0)$ is given by*

$$\tilde{\pi}(0) = [\gamma_1\pi_1 \ \gamma_2\pi_2 \ ... \ \gamma_{mK}\pi_{mK}], \tag{5.22}$$

where γ_i is given in Equation (5.21).

We note that the states $\hat{q} > m \cdot max_{th}$ are transient and the steady state probability of $\hat{q} > m \cdot max_{th}$ is 0. This can be shown as follows. From Equation (5.17) we have

$$f_i^+ = \sum_{j=\lceil \frac{i}{m} \rceil}^{K} \pi_i(j)(mj - i). \tag{5.23}$$

Since the packet drop probability is 1 for $\hat{q} \geq m \cdot max_{th}$, the steady state probability, $\pi_i(j)$, becomes

$$\pi_i(j) = \begin{cases} 1 & j = 0; \\ 0 & j > 0, \end{cases} \qquad \forall i \geq m \cdot max_{th}. \tag{5.24}$$

This is true since all the packets arriving at the queue would be dropped and the queue would be empty. Since, $\pi_i(j)$ is as given in Equation (5.24), and $i \geq m \cdot max_{th} > 0$, we get that $f_i^+ = 0, \quad \forall i \geq m \cdot max_{th}$. From Equation (5.19) we get $f_i^- > 0 \quad \forall i$. From Equation (5.21) it follows that $\gamma(i) = 0 \quad \forall i \geq m \cdot max_{th}$.

Remark 5.3.1 *The stationary probability $\pi_i(j)$ is the steady state probability of an arrival finding j customers in a $G/M/1/K$ queue. These probabilities can be computed recursively. We note that in the limit $\alpha \to 0$, the instantaneous queue moves on a much faster time scale than the estimated average queue length, and hence approaches stationarity. Therefore, if we had exponential inter-arrival times and general service times (i.e., $M/G/1/K$) instead of general inter-arrival times and exponential inter-arrival times (i.e., $G/M/1/K$) we could use the (time-average) steady state distribution of the $M/G/1/K$ system to compute the performance measures. In the limiting case, the performance measures of both $G/M/1/K$ and $M/G/1/K$ queues with RED queue management scheme could be obtained using this approach. However, for the general case ($\alpha \nrightarrow 0$), using our method, we can only obtain the performance measures of the $G/M/1/K$ queue with RED queue management scheme.*

If the arrival process to the queue is Poisson with rate λ and the service times are exponentially distributed with mean μ, then the stationary distribution of A_i, π_i, is given by

$$\pi_i := \{\pi_i(j)\} = \pi_i(0)\rho_i^j, \tag{5.25}$$

where

$$\pi_i(0) = \left(\sum_{j=0}^{K} \rho_i^j \right)^{-1},$$

and $\rho_i = \lambda \cdot (1 - h(\hat{q}_n = i)) \cdot \mu^{-1}$. Here $h(\cdot)$ is the given drop function.

From the above analysis, using Equation (5.21), we are able to obtain the stationary distribution of the average queue length variable, $\hat{q}$. We use the PASTA property which states that the arrivals see this average queue length distribution. Hence, we can calculate the average packet drop probability as

$$P_{avg} = \sum_{i=0}^{mK} \gamma(i)h(i), \tag{5.26}$$

For batch arrivals, we make the assumption that every packet in a batch is dropped with same drop probability. With this assumption, we can write the probability of dropping at least one packet in batch as

$$P_{avg} = \sum_{i=0}^{m(K-N)} (1 - (1 - \gamma(i))^N) \cdot h(i) + \sum_{i=m(K-N)+1}^{mK} h(i), \tag{5.27}$$

5.3.2 An approximate analysis

We present an approximate analysis which allows us to obtain the mean value of $\hat{q}$ without having to find the distribution.

An approximate fixed point equation. The expected value of $\hat{q}$, $\hat{q}^*$, can be obtained by solving

$$\hat{q}^* = m\mathrm{E}[Q_{\hat{q}^*}], \tag{5.28}$$

where $\mathrm{E}[Q_{\hat{q}^*}]$ is the expected queue length when packets are dropped with a constant probability $h(\hat{q}^*)$.

Using Equation (5.17) and Equation (5.19) we get

$$f_i^+ - f_i^- = m\sum_{j=0}^{K} j\pi_i(j) - i = m\mathrm{E}[Q_i] - i, \tag{5.29}$$

where $\mathrm{E}[Q_i]$ is the expected queue length in the unperturbed Markov chain with $\hat{q}_n = i$. The term $f_i^+ - f_i^-$ can be thought of as an indicator for transitions of the average queue length, $\hat{q}$. We note that in states where $m\mathrm{E}[Q_i] > i$, $f_i^+ - f_i^-$ is positive, hence indicating a tendency to increase $\hat{q}$. Similarly, when $m\mathrm{E}[Q_i] > i$, $f_i^+ - f_i^-$ is negative, hence indicating a tendency to decrease $\hat{q}$. The expected value of $\hat{q}$ will then occur at the point where $f_i^+ - f_i^- = 0$. Hence, $\hat{q}^*$ can be obtained by solving Equation (5.28). For the existence and uniqueness of the solution, we assume that the drop probability function (this is the case in the gentle variant of RED [Flo00]) is a continuous non-decreasing function, $h(\hat{q})$, of the estimated average queue length. The effective offered load, ρ^*, for a given value of $\hat{q}^*$ is then given by

$$\rho^* = \rho(1 - h(\hat{q}^*)). \tag{5.30}$$

Since ρ^* is a continuous non-increasing function on $\hat{q}^*$, we can express $\hat{q}^*$ as

$$\hat{q}^* = h^{-1}\left(1 - \frac{\rho^*}{\rho}\right) := g(\rho^*). \tag{5.31}$$

We note that g is continuous non-increasing function on ρ^*. Since the expected queue length, $\mathrm{E}[Q(\rho^*)]$, in the stationary regime is a continuous non-decreasing function of the offered load, then we can say that $\mathrm{E}[Q_{\hat{q}^*}]$, is a continuous non-increasing function of $\hat{q}^*$. As $\hat{q}^*$ and $E[Q(\cdot)]$ map to the same interval, $[0, K]$, we can say that the solution to Equation (5.28) exists and is unique.

Remark 5.3.2 *The fixed point equation was also obtained in [SVL02] as the stationary solution of an ordinary differential equation.*

5.3.3 The general case

In Section 5.3.1 we presented an analysis for the limiting case $\alpha \to 0$. In this section we outline an algorithm with which we can compute the steady state probabilities when $\alpha \nrightarrow 0$. However, α still has to satisfy the condition $\alpha mK < 1$.

The transition probability matrix of the two dimensional Markov chain $(\hat{q}_n, Q_n)$ given in Equation (5.11) can be written as

$$\tilde{P} = \begin{bmatrix} L_0 & F_0 & 0 & \dots & \dots \\ B_1 & L_1 & F_1 & 0 & \dots \\ 0 & B_2 & L_2 & F_2 & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & B_{mK} & L_{mK} \end{bmatrix}, \tag{5.32}$$

where B_i, L_i, F_i denote the backward, local, and forward transition matrices, respectively. This is the transition matrix for a level dependent QBD process. Let the $\tilde{\pi}$ be stationary distribution of $\tilde{P}$. Let $\tilde{\pi} = [\tilde{\pi}_0 \tilde{\pi}_1 ... \tilde{\pi}_{mK}]$, where $\tilde{\pi}_i$'s are vectors of size $1 \times K$. $\tilde{\pi}_i$ is the steady state probability vector of Q for $\hat{q} = i$. $\tilde{\pi}$ can be found using the following algorithm [GJL84].

1. Compute the S_i matrices using the following recursion

$$\begin{aligned} S_0 &= L_0 \\ S_i &= L_i + B_i(I - S_{i-1})^{-1} F_{i-1}, \qquad 1 \le i \le mK. \end{aligned}$$

2. $\tilde{\pi}_{mK}$ is computed by solving

$$\begin{aligned} \tilde{\pi}_{mK} &= \tilde{\pi}_{mK} S_{mK}, \\ \tilde{\pi}_{mK} \cdot \underline{1} &= 1. \end{aligned}$$

3. $\tilde{\pi}_i$, $0 \le i \le mK$ is computed using the recursion

$$\tilde{\pi}_i = \tilde{\pi}_{i+1} B_{i+1} (I - S_n)^{-1},$$

4. $\tilde{\pi}$ is found by normalization

$$\tilde{\pi} = \frac{\tilde{\pi}}{\tilde{\pi} \cdot \underline{1}}$$

5.4 Numerical results

In this section we present the results obtained through the analysis as described in the previous section and the results obtained through simulations. Our aim to see the effect of the approximations that we made in the model and to verify that these effects are indeed negligible. The arrival process is assumed to be Poisson batch arrival process with rate λ and fixed batch sizes, N. The service times are assumed to be exponential with mean $1/\mu$. The offered load, ρ, is defined as $\rho = \frac{\lambda N}{\mu}$.

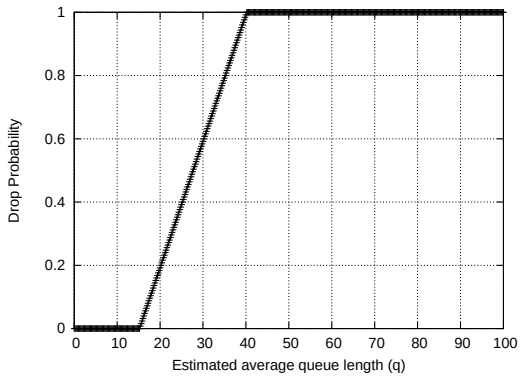


Figure 5.3: RED drop probability function

Remark 5.4.1 *The validity of exponential inter-arrival time has been widely debated. Various authors have shown that for different scenarios the traffic is self-similar and not Poissonian [LTWW94], [PF95], [CB97]. However, in [CCLS03], the authors argue that under heavy load conditions the traffic tends towards a Poisson process. In a recent article [KMFB04], the main findings suggest that, when the level of multiplexing is large, packet arrivals appear Poissonian at small time scales, non-stationary at medium time scales, and long-range dependent at large time scale. Thus, as a preliminary study, we present some results related to the performance of a RED queue when the inter-arrival time is exponentially distributed.*

The analytical results were obtained by numerically calculating the values using MATLAB. The buffer length, K, is taken to be 100 whereas the discretization parameter, m, is taken to be 5. The drop probability function is taken to be of the form

$$P_{drop} = \begin{cases} 0, & \hat{q} < min_{th} \\ \frac{\hat{q} - min_{th}}{max_{th} - min_{th}}, & min_{th} \leq \hat{q} \leq max_{th} \\ 1, & \hat{q} > max_{th} \end{cases} \tag{5.33}$$

We note that all the values of min_{th}, max_{th}, and $\hat{q}$ are scaled by a factor m in the analysis in order to discretize the estimated average queue size. The unscaled values for min_{th} and max_{th} are 15 and 40, respectively, and the drop probability function versus the unscaled estimated average queue length is as shown in Figure 5.3. We shall use the average drop probability and the probability of at least one drop in a burst as performance measures. The simulations were performed using a program written in C language [KR88], and approximately 10^8 packets were generated during the simulations.

5.4.1 The limiting case

First, we present the results for the limiting case. In the simulations, the averaging parameter, α, was taken to be 0.0001. In Figure 5.4, we plot the analytical as well as the simulation results for average packet drop probability as a function of the offered load for different values of burst sizes. There is a fairly good correspondence between the analytical and simulation results.

In [BMB00] it has been argued that the number of consecutive drops becomes infinite with positive probability as $\alpha \to 0$. However, for the drop function as shown in Figure 5.2, the number of consecutive drops appears to follow a geometric behaviour for small values of the number of

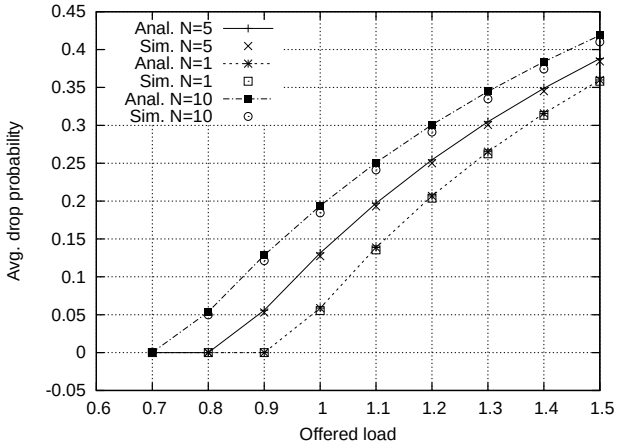


Figure 5.4: Average packet drop probability versus offered load

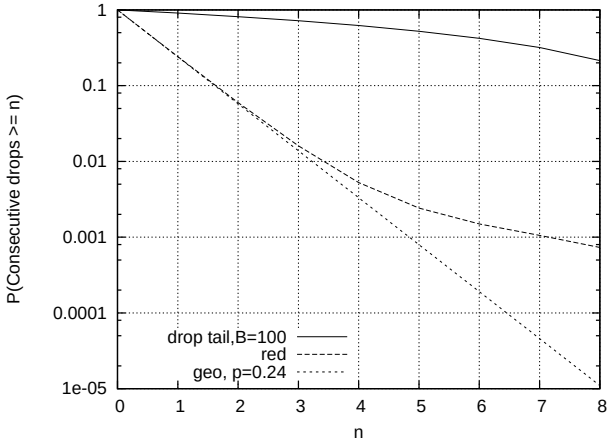


Figure 5.5: Probability of greater than or equal to n consecutive drops. $\rho = 1.1$. Burst size = 10.

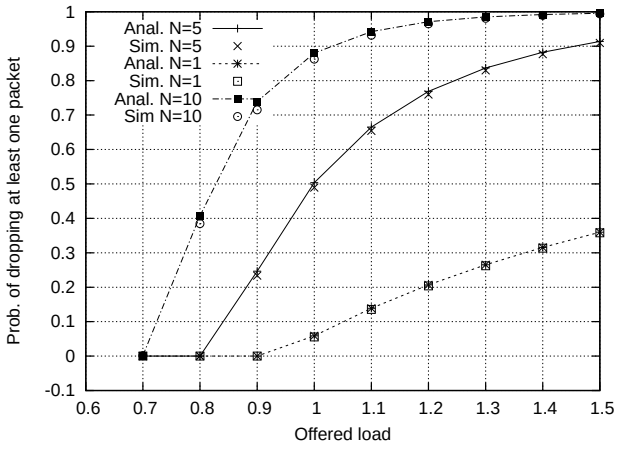


Figure 5.6: Probability of drop of at least one packet in a burst versus offered load

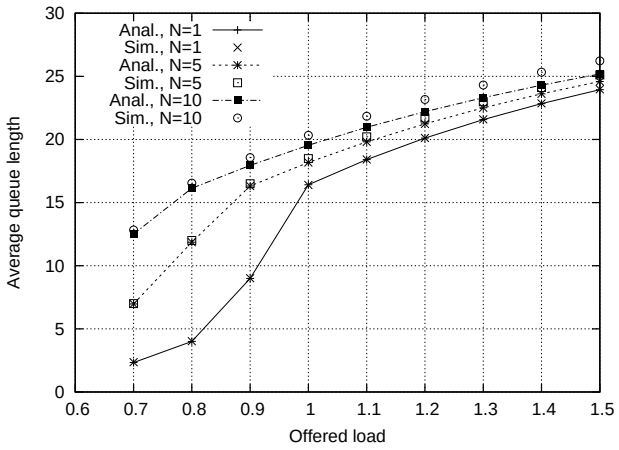


Figure 5.7: Average queue length versus offered load

consecutive drops. We plot the probability that there are greater than or equal to n consecutive drops as a function of n for a drop tail and a RED queue in Figure 5.5. The decay rate of the tail is much faster in the case of a RED queue when compared to that of a drop tail queue. For small values of n, the decay rate is close to p^n, where p is the average packet drop probability. Most of the mass can be seen to be concentrated within small values of n and, so the geometric approximation appears to be a reasonable one. A continuous drop function ensures that the drops are independent and, thus, reduces the probability of bursty losses. This argument supports using gentle RED. The tail drop probability during this simulation scenario was 2×10^{-4} which also suggests that synchronised losses are reduced.

As mentioned above, the probability of dropping consecutive packets becomes independent when the drop function is continuous. This suggests that the probability of drops in a burst is binomially distributed. The probability of at least one drop in a burst can thus be approximated by $1 - (1 - p)^N$, where p is the average drop probability and N is the burst size. This behaviour is seen in Figure 5.6 in which the probability of dropping at least one packet in a burst is plotted as a function of the offered load.

Next, we validate the approximate analysis for obtaining the average queue length. In order to obtain the average queue length we need to solve the fixed point equation as given in Equation (5.28). For example, the expected queue length as observed by a incoming packet in a $M/M/1$ queue with finite buffer K is given by

$$\mathrm{E}[Q_{\hat{q}^*}] = \frac{\rho_{\hat{q}^*}}{(1-\rho_{\hat{q}^*})} - (K+1)\frac{{\rho_{\hat{q}^*}}^{K+1}}{(1-{\rho_{\hat{q}^*}}^{K+1})}, \tag{5.34}$$

where

$$\rho_{\hat{q}^*} = \rho\left(1 - \frac{\hat{q}^* - min_{th}}{max_{th} - min_{th}}\right).$$

We can now obtain the expected value of average queue length by solving

$$\hat{q}^* = \frac{\rho_{\hat{q}^*}}{(1-\rho_{\hat{q}^*})} - (K+1)\frac{{\rho_{\hat{q}^*}}^{K+1}}{(1-{\rho_{\hat{q}^*}}^{K+1})}. \tag{5.35}$$

Similarly, for the batch arrivals we use the expression for the infinite buffer case as an approximation. The expected value, $\hat{q}^*$, for batch arrivals is obtained by solving

$$\hat{q}^* = \frac{\rho_{\hat{q}^*}}{(1-\rho_{\hat{q}^*})}\frac{N+1}{2}. \tag{5.36}$$

For the values of min_{th}, max_{th}, and K assumed in this section, we plot in Figure 5.7 the average queue as obtained from solving Equation (5.35) and Equation (5.36) numerically using MATLAB and the results obtained through simulations.

5.4.2 The general case

Next, we consider the case when $\alpha \nrightarrow 0$. In the rest of this section we assume that the discretization parameter, m, is 1. The drop function is the same as in the previous section. We need that $\alpha m K$ be less than 1. Thus, α has to be chosen such that $\alpha < 0.01$ in order to use the QBD algorithm. We choose $\alpha = 0.009$ which is close to 0.01. In Figure 5.8 and Figure 5.9 we plot the average drop probability and the average queue length, respectively, as a function of the offered load. We note that the behaviour is similar to that observed in the limiting case.

In order to see the effect of choosing an averaging parameter which is not necessarily small, we plot in Figure 5.10 the distribution function of the average queue length, $\hat{q}$, using the QBD analysis, simulations, and the singular perturbation (SP) analysis. The offered load was 1.2 and the burst size was taken to be 10. As can be observed from the figure, the effect of increasing α is to increase the variance of $\hat{q}$. We cannot, however, reduce the variance of $\hat{q}$ by taking the limit. There is a limiting distribution of $\hat{q}$ (as was seen from the limiting case) and hence a limiting variance which is different from 0.

5.5 Summary and conclusions

We used a singular perturbation based approach to find the limiting stationary distribution of the average queue length and the packet drop probability in a RED enabled queue. For Poisson arrival and exponential service times the limiting stationary distribution was given as closed form expressions. The expected value of the average queue length was found to be the fixed point solution

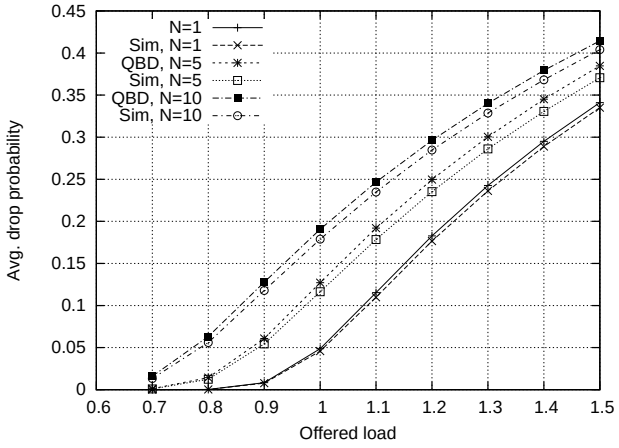


Figure 5.8: Average drop probability versus offered load.

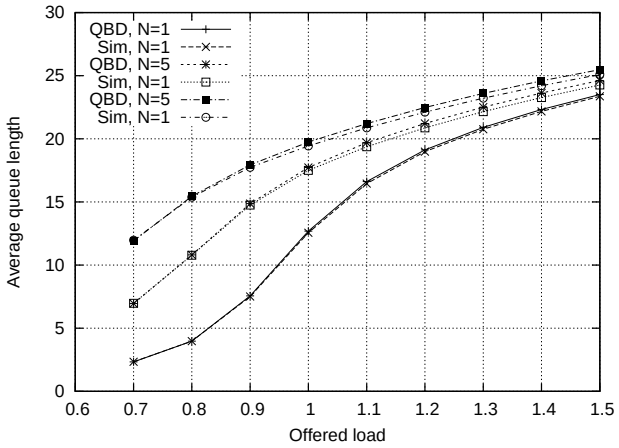


Figure 5.9: Average queue length versus offered load.

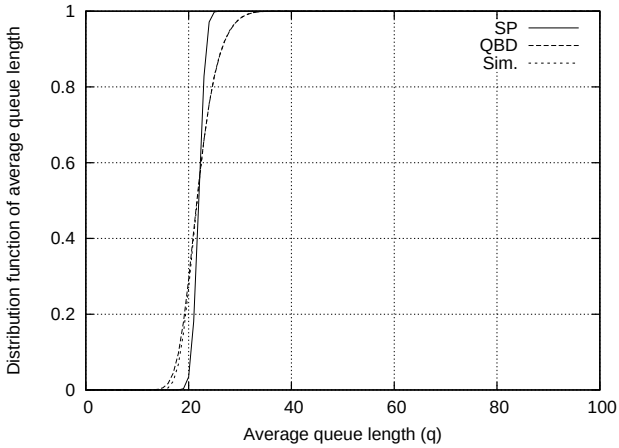


Figure 5.10: Distribution function of the average queue length. $N = 10$ and offered load is 1.2. $\alpha = 0.009$.

of a function depending on the average queue length of a system with no averaging. The analytical results were observed to match fairly accurately with results obtained through simulations. We also showed that by using a continuous drop function we can reduce the number of consecutive losses thereby avoiding synchronised losses. We showed that for the case when the averaging parameter is not small, the stationary distribution can be computed using an algorithm for level dependent QBDs. However, for small values of the averaging parameter, the computational complexity of using this algorithm was much greater than that based on the singular perturbation approach. In this work, we considered the scenario in which packet generation process is independent of packet drops. Future work will involve the study of a RED queue with closed loop TCP traffic sources.

Chapter 6

Optimal control of delay and energy in a wireless device

In the second part of the thesis, we shall consider two problems related to wireless devices and wireless networks. The first problem that we study is related to optimizing the energy-delay tradeoff faced by a wireless device. Specifically, we study a decision problem faced by an energy limited wireless device that wants to transmit data and operates in discrete time. There is some external arrival to the device's transmit buffer. The possible decisions are: (*i*) to serve some of the buffer content; (*ii*) to reorder a new battery after serving the maximum possible amount that it can; or (*iii*) to remain idle so that the battery charge can increase owing to diffusion process (which is possible in some commercially available batteries). We shall formulate the decision problem in the framework of Markov Decision Processes. We shall address both finite and infinite horizon discounted costs as well as average cost minimization problems. Without using any second order characteristics, we obtain results that include: (*i*) optimality of bang-bang control; (*ii*) the optimality of threshold based policies; (*iii*) parametric monotonicity of the threshold; and (*iv*) uniqueness of the threshold.

The results in this chapter have appeared in [BKP05].

6.1 Introduction

Wireless devices are constrained in their operational lifetime by finite energy batteries. Therefore, energy efficient design of protocols at different layers of the protocol stack for wireless networks has recently received significant attention, see, for example, [CGB04], [YHE02], [ZR01] and [PSAC01]. Although the primary objective of a device is to transmit and receive data with minimal delay, this must be done with the added constraint of minimizing the transmission costs and increasing the operational lifetime of the device. In [GKS03], the authors studied delay optimal packet scheduling policies subject to average transmit power constraint over a wireless channel with independent fading. In [RB05], the authors extended this model to include Markovian fading. Although in the above mentioned articles an average transmit power constraint was imposed, an interesting feature of the battery was ignored. In [FDN94], it was observed that a battery when left idle can regain some of its lost charge. This phenomenon, known as relaxation phenomenon, allows a battery operated in pulsed (or intermittent) discharge mode to deliver more energy than the same battery

operated in continuous discharge mode. This enables a user to send more packets and increase the operational lifetime of the battery by leaving it idle in between packet transmissions, thus providing an incentive to remain idle even though the transmit buffer may not be empty. However, this would add to the delay of the packets queued up in the buffer. This trade-off between energy and delay leads to a decision making problem formulation where the user has to decide whether to serve packets or leave the terminal idle in order to minimize certain costs.

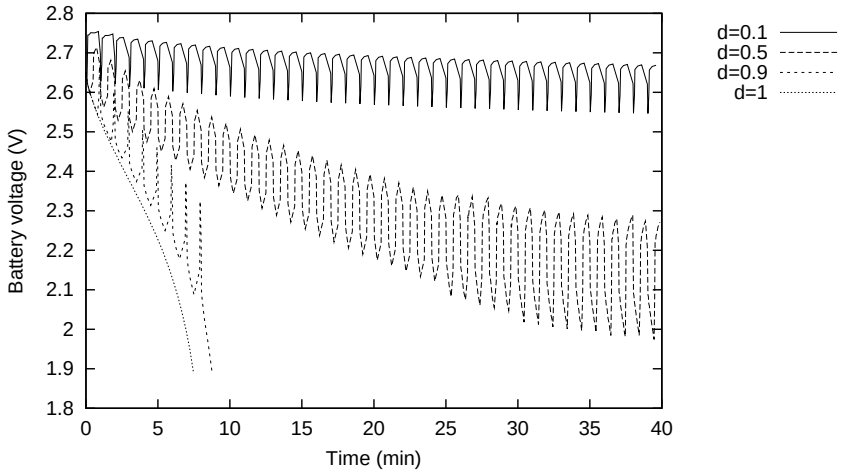


Figure 6.1: An illustration of the relaxation phenomenon.

In Figure 6.1[1], we illustrate the effect of pulsed discharge on the lifetime of a Lithium ion cell. The data has been obtained using a computer program which simulates battery behaviour [New]. The parameter d represents the ratio of total ON time to the total time. By reducing the value of d, i.e., by increasing the idle time, the operation lifetime of the cell increases. Although the time scale on the graph is large (of the order of minutes), a similar phenomenon has been also observed on a time scale of tens of milliseconds [PSS01]. The advantages of relaxation phenomenon in the context of wireless devices was first presented in [CR99]. Since then several authors have proposed algorithms which take advantage of the relaxation phenomenon. In a series of papers-[CR01a] and [CR01b], the authors propose a stochastic model of a battery taking into account the relaxation phenomenon. A discrete-time system is considered where, in every time slot the battery is idle, it recovers a unit of charge with some probability. The performance measure of interest is the gain in the operational lifetime. They propose a threshold based scheme wherein the transmission of packets is interrupted when the battery charge falls below a given threshold. This scheme results in a tradeoff between delay and energy efficiency. However, they do not consider an optimization framework with costs for packet delay and battery charge. In [CR01b], the problem of choosing a cell among a pack of cells is studied and the gain in operational lifetime for various schemes such as round-robin, random selection, and best-of-the-pack scheme is computed. A battery aware Medium Access Control (MAC) algorithm is proposed in [JMM04]. In this algorithm, a node uses the battery state information of its two hop neighbours in order to schedule packets. In [PSS01], the authors study the effect on a lithium ion cell when a sensor is operated in sleep mode. They observe that gains of over a 100% can be realised by operating a sensor alternatingly in sleep mode and in receive mode.

[1]This figure has been reproduced from [Pra02].

In this chapter we aim to provide a framework to quantify the trade-off between energy efficiency and delay. Unlike in [CR01a] where the main consideration is energy efficiency, we impose costs on both delay and energy consumption. We consider a discrete time system in which a user with a finite energy battery terminal has to decide whether to serve packets or to leave the system idle in each time slot. Further, the user can decide to replace the battery with a new one at an additional cost. We note that there are two variables (i.e., energy level of the battery and the length of the transmit buffer) based on which a decision is to be made. We formulate the problem as a Markov decision process. We then derive the structural properties based on the directional derivative of the value function. We first consider a finite horizon problem and provide the structure of the optimal policy. We then extend this to the infinite horizon discounted cost problem, and finally consider the infinite horizon average cost minimization.

6.2 Outline

In Section 6.3, we shall formulate the energy-delay trade-off as an optimization problem. We shall relate the energy-delay trade-off problem to a problem of optimal control of a discrete time tandem queue. Since these two problems are equivalent, we shall solve the wireless device problem by solving the tandem-queue problem. First, in Section 6.4, we shall consider the finite horizon discounted cost problem and obtain structural results. These results show the possibility of two different optimal policies depending on the derivatives of the cost functions. The two possibilities will be studied analytically in Sections 6.5 and 6.6, where we shall also give a numerical example to validate the results. The summary and conclusions will be presented in Section 6.7.

6.3 Problem formulation

We consider the optimal control problem where, at the beginning of each time slot (i.e., a decision epoch), the user is aware of the current buffer occupancy and the energy level of the battery, and hence can take an action based on these two parameters. However, it does not have any knowledge of the amount of data that will arrive in the transmit buffer during the current time slot. In this section, we shall formalize the problem statement mentioned in Section 6.1.

Some words on the notation: We shall use the notation z^+ to denote $\max(z, 0)$, $\mathrm{sg}(x)$ to denote the sign of x, and $\mathbf{I}(a)$ to denote the indicator function of an event a.

Let x_n and p_n denote the buffer length and the remaining energy level, respectively, at the beginning of the nth time slot. We assume x_n is infinitely divisible and $x_n \in [0, \infty), \forall n$, i.e., the buffer content is fluid and there is infinite buffer space. The remaining energy level, p_n, is assumed to be bounded above by M, i.e., $p_n \in [0, M], \forall n$. A linear relationship is assumed between the amount of energy and the amount of fluid that can be transmitted using this energy (see, for example, [Bor03] where the throughput is assumed to be a linear function of the transmit power). In other words, a unit of energy is required to serve each unit of fluid. The state space, $\mathcal{C}$, is given by $\mathcal{C} = \{(x, p) : x \in [0, \infty), p \in [0, M]\}$. The cost function associated with the state (x, p) is denoted by $h(x) + g(p)$, where $h(x)$ is a non-negative and increasing function of x, and $g(p)$ is a non-negative and decreasing function of p. Let w_n denote the amount of fluid which arrives during the nth time slot. The random sequence $\{w_n\}, n \geq 0$, is assumed to be independent and identically distributed with distribution function $\mu(\cdot)$.

We assume that in every slot in which the battery is left idle, the residual battery energy increases from p to some amount $p + \zeta(p) \geq p$. In the rest of this chapter, we will drop the

dependence on p of $\zeta(p)$ and use ζ to denote the function. We note that the case $\zeta = 0$ corresponds to the other practical scenario where the battery does not gain any charge when left idle. In the state (x, p), the user can take one of the following actions:

1. remain idle[2],
2. serve some amount $u \in [0, \min(x, p)]$, or
3. serve $\min(x, p)$ and reorder a new battery with residual energy level M.

A cost of $r(p)$, where $r(\cdot)$ is a non-decreasing function of p, is incurred each time a battery is reordered in state (x, p). A policy π defines an action for each $(x, p) \in \mathcal{C}$. Let $\beta < 1$ denote the discount factor. We study optimal policies which minimize one of the following three cost criteria.

– Finite horizon discounted cost:

$$E \sum_{k=0}^{N} \beta^k (h(x_k) + g(p_k) + r(p_k)\mathbf{I}(\text{reorder})),$$

for some $N > 0$.

– Infinite horizon discounted cost:

$$E \sum_{k=0}^{\infty} \beta^k (h(x_k) + g(p_k) + r(p_k)\mathbf{I}(\text{reorder})).$$

– Infinite horizon average cost:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N} (h(x_k) + g(p_k) + r(p_k)\mathbf{I}(\text{reorder})).$$

Remark 6.3.1 *In this model, we have not taken into account certain aspects of the relaxation phenomenon (see, for example, [CR01a]). Firstly, the amount of recovered energy when the battery is left idle depends not only on the present residual charge but also on the entire discharge/recharge history which may not easy to model and analyse. Secondly, the maximum amount of energy that a battery can deliver is finite. In our formulation, the user can take decisions so that the battery can recover charge an infinite number of times, i.e., it could use one battery forever. However, as a first approximation, we analyse a simplified model.*

6.3.1 An equivalent discrete-time tandem queue

Let us consider a discrete-time tandem queue with one fixed server and one moving server as shown in Figure 6.2. A controller performs the task of scheduling the two servers. In each slot, the moving server can serve either of the two queues. The fixed server can serve only the second queue, and it can do so only when the moving server is at the first queue. That is to say, both the servers cannot

[2]By remaining idle we mean going into sleep mode. The battery can recover some charge even though a small current is drawn to keep the system in sleep mode [PSS01].

serve the second queue in the same slot. However, it is possible that the moving server serves the first queue and the fixed server serves the second queue in the same slot.

During each slot, a random amount of work arrives in the first queue. The first queue has an infinite buffer capacity whereas the second queue's buffer capacity is finite. The work served from the first queue enters the second queue. There is a further source of strong interaction between the two queues: the amount of service imparted to the first queue in any slot cannot exceed the remaining buffer capacity of the second queue at the beginning of the slot (this is to ensure that no work is lost). This is the case when the first queue's server (i.e., the moving server) does not take into account the fact that the work in the second queue may have been depleted because of some service imparted by the fixed server. There is a workload-dependent holding cost associated with each of the two queues.

In one slot, the moving server can perform either (at most) M units of work on the first queue or (at most) $\zeta(y) \leq M$ units of work on the second queue (i.e., the work done can depend on the workload, y). In one slot, the fixed server can perform at most M units of work on the second queue. However, there is a fixed cost, c, incurred in every slot in which work is done by the fixed server whereas there is assumed to be no cost associated with the service performed by the moving server. Since the cost c is independent of the work done by the fixed server, it is optimal for the fixed server to exhaustively serve the second queue. At the beginning of each slot, the controller has to decide whether to activate the fixed server or not, and the amount of work to be done on each queue. The objective of the controller is to minimize the long term average of the cost incurred in each slot.

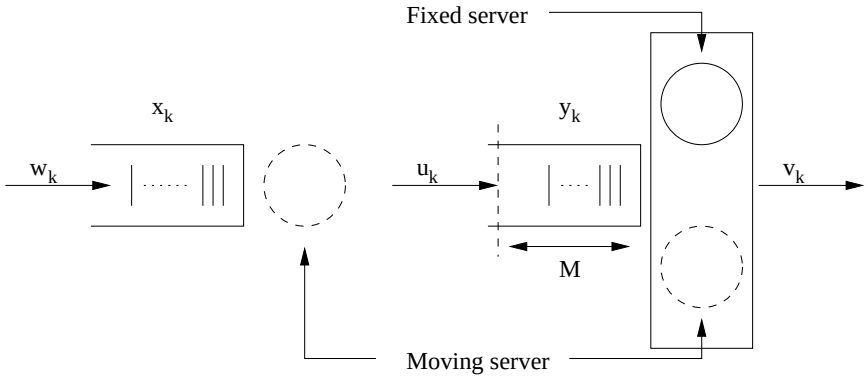


Figure 6.2: The discrete-time tandem queue model.

6.3.2 Analogy between the wireless problem and the tandem queue problem

The first queue in the tandem queue problem is analogous to the transmit buffer of the wireless problem, while the second queue is analogous to the battery. The analogy between the second queue and the battery is better understood by using the relation $y_k = M - p_k$. The workload in the second queue corresponds to the amount of energy consumed from the battery. After the moving server serves u_k amount of work from the first queue (i.e., u_k amount of fluid is transmitted from the transmit buffer), this work enters the second queue (i.e., the energy level reduces by u_k). If the moving server serves $\zeta(y_k)$ amount of work from the second queue, it corresponds to an increase in the residual energy level by $\zeta(M - p_k)$. The constraint that the moving server can only serve one queue in one slot corresponds to the constraint that either fluid can be transmitted from the

transmit queue or the residual charge in the battery can increase. The service imparted by the fixed server corresponds to replacing the battery.

With the help of the analogy described above, we can solve the wireless problem by solving the tandem queue problem. In the following subsection, we shall formulate the tandem queue problem as a Markov decision problem. We shall mention a few articles relevant to the tandem queue problem. In order to solve the Markov decision problem, we shall use the vanishing discount approach to study the average cost problem. Specifically, we will first study the infinite horizon discounted cost dynamic programming problem and then let the discount factor approach unity. The finite storage capacity of the second queue leads to non-convexity of the value function. Therefore, convexity preserving arguments cannot be used to demonstrate the optimality of a threshold type control policy. However, we provide conditions for the cost functions for which a bang-bang policy at the first queue and a threshold policy at the second queue are optimal.

6.3.3 Dynamic programming problem formulation

We consider a tandem queue (as shown in Figure 6.2) which is operated in discrete time. Let x_k and y_k denote respectively the workload in the first queue and in the second queue at the beginning of slot k. The state space of the system can be described by a two-dimensional vector (x_k, y_k) where $x_k \in [0, \infty)$ and $y_k \in [0, M]$. The cost function associated with state (x_k, y_k) is given by $h(x_k) + g(y_k)$ (the corresponding cost function for the wireless problem is $h(x) + g(M - p_k)$. The functions $h(\cdot)$ and $g(\cdot)$ are assumed to be positive and increasing. Let w_k denote the amount of work that arrives in the first queue during slot k. The random variable w_k is assumed to be i.i.d. with distribution function $\mu(\cdot)$ and mean $\mathrm{E}[w] < M$. The state equations can be written as

$$\begin{aligned} x_{k+1} &= x_k + w_k - u_k, \\ y_{k+1} &= y_k + u_k - v_k. \end{aligned} \tag{6.1}$$

Due to the finite storage capacity of the second queue, the amount of work done on the first queue (i.e., u_k) depends on the work present in the second queue, i.e., $u_k \in [0, \min(x_k, M - y_k)]$. Moreover, if the fixed server is activated and the mobile server serves some amount of work at the first queue, then the fixed server can serve this work at the second queue in the same slot i.e., $v_k \in [0, \min(y_k + u_k)]$. Let $\phi_k \in \{0, 1\}$ denote whether the fixed server is inactive or active in the kth slot. In each slot in which $\phi_k = 1$, a fixed cost of c is incurred. In each slot, the controller can take one of the following actions.

1. To activate the fixed server, i.e., $\phi_k = 1$, in which case it is optimal for the moving server to serve the maximum possible in the first queue, i.e., $u_k = \min(x_k, M - y_k)$.

2. To disable the fixed server, i.e., $\phi_k = 0$, in which case the moving server could either serve $u_k \in [0, \min(x_k, M - y_k)]$ at the first queue or serve $v_k = \min(y_k, \zeta)$ at the second queue

Remark 6.3.1 *1. Since a fixed cost is incurred independently of the amount served when the fixed server is active, and the fixed server can serve at most $y_k + u_k$, it is optimal for the moving server to maximize u_k.*

2. Since the work done on the first queue enters the second queue, the controller has to decide on the amount of work the moving server does on the first queue (i.e., $u_k \in [0, \min(x_k, M - y_k)]$) when $\phi_k = 0$. There is a trade-off between reducing the work (and, hence, the holding cost)

in the first queue and increasing the work (and, hence, the holding cost) in the second queue. On the other hand, service at the second queue reduces the overall holding cost. Therefore, the moving server serves the maximum possible amount at the second queue (i.e., $\min(y_k, \zeta)$*).*

The action performed in each slot is a 3-tuple (ϕ_k, u_k, v_k). The set of actions, $\mathcal{A}$, available to the controller contains the 3-tuples

$$\mathcal{A} = \begin{cases} (1, \min(x_k, M - y_k), \min(x_k + y_k), M), & \text{(6.2a)} \\ (0, u_k, 0), \quad u_k \in [0, \min(x_k, M - y_k)], \text{ and} & \text{(6.2b)} \\ (0, 0, \min(y_k, \zeta)). & \text{(6.2c)} \end{cases}$$

Remark 6.3.2 *1. The action (2b) is itself an uncountable set of actions because the controller has to decide on the amount of work to be served in the first queue.*

2. For $\phi_k = 0$*, the action set contains the possibility of the mobile server not serving the second queue even though the first queue is empty, and vice versa (i.e., the action could be* $(0, 0, 0)$ *even though the state* (x_k, y_k) *may not be* $(0, 0)$*). Later we shall see that it is not optimal to take this action if the state is not* $(0, 0)$*.*

In order to study the infinite horizon discounted cost problem, one usually looks at the value iteration, thus effectively solving the finite horizon discounted cost problem as well [Put94]. Hence, we study the optimal policy for the three cost criteria in the following order.

– Finite horizon discounted cost (FHDC):

$$\text{minimize} \quad E \sum_{k=0}^{N} \beta^k (h(x_k) + g(y_k) + c \cdot \mathbf{I}(\phi_k = 1)), \tag{6.3}$$

for some $N > 0$ and a discount factor β.

– Infinite horizon discounted cost (IHDC):

$$\text{minimize} \quad E \sum_{k=0}^{\infty} \beta^k (h(x_k) + g(y_k) + c \cdot \mathbf{I}(\phi_k = 1)), \tag{6.4}$$

for some $\beta < 1$.

– Infinite horizon average cost (IHAC):

$$\text{minimize} \quad \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N} (h(x_k) + g(y_k) + c \cdot \mathbf{I}(\phi_k = 1)). \tag{6.5}$$

We assume that the controller is aware of the state of the system and hence can decide based on this information. This problem can be formulated as the above described discrete-time Markov decision process for which we provide the conditions for optimality of bang-bang control.

6.3.4 Literature related to control of discrete-time tandem queues

We discuss two articles that are related to this work. One study of control of tandem queue that appears to be very similar to our problem is that of [ZVW82] (they consider linear cost functions). However, they assume $\zeta = 0$ (i.e., the moving server is fixed at the first queue), and that there is always a dummy customer present in the second queue so that the term corresponding to our $(x - M)^+$ term does not show up in their case; this is also the reason that they are able to show convexity of the value function. This approach can be taken in their model as they assume Poisson arrivals and exponentially distributed service requirements. We can not do this since we are working with a discrete time queue. Yet another study is of [Haj84] which considers two interconnected queues and shows that the optimal policy is that of a monotone switching curve.

These studies are different from the present work in at least the following aspects:

- Since [ZVW82] and [Haj84] consider systems in continuous time under Poissonian assumptions, the jump in any dimension of the state can be at most by one (an arrival or a departure). However, we are considering a discrete time system, hence we are allowing for a general arrival process (and hence, jumps in state space).

- In both the above mentioned works ζ is assumed to be zero, i.e., the moving server is fixed at the first queue.

- Both the above mentioned studies consider a linear holding cost functions. We are allowing for a more general objective function, and in fact *we will characterize a wider class of holding cost functions* for which threshold based policies are optimal. Moreover, we will not use strict second order characteristics of the holding cost functions and will establish important structural results like parametric monotonicity for the mentioned class of holding cost functions.

- We are assuming that the buffer capacity of the second queue is finite. As mentioned already, this results in a strong dependence of the amount of service to the first queue. It turns out that the characterization of the value function in the infinite horizon discounted cost problem depends crucially on the buffer capacity of the second queue. In particular, starting with the same holding cost functions $h(\cdot)$ and $g(\cdot)$, the IHDC value function is M-convex (we shall define M-convexity in Section 6.5) and hence depends on the buffer capacity M of the second queue. This characterization of the IHDC value function is also a new contribution that the present work makes.

A survey of control of service rate in a network of queues is presented in [SW93].

6.4 The finite horizon discounted cost problem

Let $V_k(x, y)$ denote the value function when the state is (x, y) and there are k more decision epochs (i.e., k more slots) remaining before reaching the horizon (i.e., we terminate the decision making process). We shall assume that $V_0(x, y)$ is equal to zero for all (x, y). The dynamic programming

equation for the optimal value function is

$$V_{k+1}(x,y) = h(x) + g(y) + \beta \min \big\{ E_w V_k((x-(M-y))^+ + w, 0) + c, \\ \min_{0\le u\le \min(x,M-y)} E_w V_k(x+w-u, y+u), \\ E_w V_k(x+w, (y-\zeta)^+) \big\}. \tag{6.6}$$

The operator E_w is the expectation over the random variable w. The term $E_w V_k((x-(M-y))^+ + w, 0)$ corresponds to activating the fixed server and serving the maximum possible amount at the first queue (i.e., action (2a)). The term $\min_{0\le u\le x\wedge(M-y)} E_w V_k(x+w-u, y+u)$ corresponds to $\phi_k = 0$ and the mobile server serving some amount u_k from the first queue (i.e., action (2b)). The final term, $V_k(x+w, (y-\zeta)^+)$, corresponds to $\phi_k = 0$ and the mobile server serving $\min(y,\zeta)$ at the second queue (i.e., the action (2c)).

For the rest of this section, we shall assume that $\beta = 1$ with a remark that all the results of this section are valid when we consider a discount factor of $\beta < 1$. Let N denote the finite horizon. We first provide a sufficient condition under which *bang-bang* control is optimal at the first queue, i.e., the decision is to either serve the maximum possible quantity or to serve nothing. This reduces the decision space of u_k from $u_k \in [0, \min(x, M-y)]$ to $u_k \in \{0, \min(x, M-y)\}$. The condition which will allows us to reduce the decision space is quite general and is satisfied by, among others, linear functions $h(x) = c_1 x$ and $g(y) = c_2 y$. We shall also provide some structural results of the optimal policy, like parametric monotonicity for the threshold policy. We point out here that we obtain the results of this section *without using second order characteristics of the value functions*, like convexity or decreasing-differences property.

6.4.1 Optimality of bang-bang control

Let $\nabla V_k(x,y) = -\frac{\partial}{\partial a} V_k(a,b) + \frac{\partial}{\partial b} V_k(a,b)|_{a=x,b=y}$ denote the directional derivative[3] of $V_k(a,b)$ along the vector $(-1,1)$, evaluated at the point (x,y). We assume that $V_1(x,y) = h(x) + g(y)$ satisfies the following condition.

Condition A *The function $\nabla V_1(x,y) = -h'(x) + g'(y)$ has the same sign, say $\mathcal{S} \stackrel{\text{def}}{=} \text{sg}(-h'(x) + g'(y)) = \text{sg}(\nabla V_1(x,y))$, for all values of x and y.*

Note that this condition is satisfied by the linear cost functions $h(x) = c_1 x$ and $g(y) = c_2 y$. Under Condition A we have the following two lemmas.

Lemma 6.4.1 *For all values of x, y, w, and a given $k \le N$, the minimizer $u^* = \operatorname{argmin}_{0\le u\le \min(x,M-y)} E_w V_k(x+w-u, y+u)$ is given by*

$$u^* = \begin{cases} 0 & \text{if} \quad \text{sg}(\nabla V_k(a,b)) = + \quad \forall (a,b), \\ \min(x, M-y) & \text{if} \quad \text{sg}(\nabla V_k(a,b)) = - \quad \forall (a,b). \end{cases}$$

Proof. The distribution function of w is given by $\mu(\cdot)$. Therefore,

$$E_w V_k(x+w-u, y+u) = \int_w V_k(x+w-u, y+u) d\mu(w). \tag{6.7}$$

[3] We use the term derivative and right derivative interchangeably.

The derivative of $E_w V_k(\cdot,\cdot)$ with respect to u can be written as

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{du}} E_w V_k(x+w-u, y+u) &= \nabla E_w V_k(x+w-u, y+u) \\ &= \int_w \nabla V_k(x+w-u, y+u) d\mu(w).\end{aligned} \tag{6.8}$$

Since $\nabla V_k(x+w-u, y+u)$ has a sign independent of u, x, y, w, it follows that $\nabla E_w V_k(x+w-u, y+u)$ has the same sign as $\nabla V_k(x+w-u, y+u)$. Therefore, $E_w(V_k(x+w-u, y+u)$ is an increasing (resp., decreasing) function of u when $\mathrm{sg}(\nabla V_k(a,b)) = +$ (resp., $-$). The lemma thus follows. ∎

Lemma 6.4.2 *Under Condition A, $\nabla V_k(x,y)$ has the same sign as $\nabla V_1(x,y)$ for all values of $k \leq N, x,$ and y.*

Proof. We use the induction argument to prove this lemma. For $k = 1$, from Condition A, $\mathrm{sg}(\nabla V_1(x,y))$ is the same for all (x,y), hence the claim is true. Let $\mathcal{S} = +$, and let the claim be true for some $k \geq 1$, i.e., $\mathrm{sg}(\nabla V_k(x,y)) = \mathcal{S}$ for all (x,y). Using Lemma 6.4.1, and Equation 6.6 with $\beta = 1$, we obtain

$$V_{k+1}(x,y) = h(x) + g(y) + \min \big\{ E_w V_k((x-(M-y))^+ + w, 0) + c, \tag{6.9}$$

$$E_w V_k(x+w, y), E_w V_k(x+w, (y-\zeta)^+) \big\}. \tag{6.10}$$

From the hypothesis, $\mathrm{sg}(\nabla V_k(x,y)) = \mathcal{S}$ for all values of (x,y). As the expectation operator gives a convex combination, it follows from the hypothesis and envelop theorem [Ber95, Lemma 3.3.1] that each of the terms under the minimum operator above will have the same property, i.e., their directional derivatives along vector $(-1,1)$ will have sign $\mathcal{S}$. Therefore, $\mathrm{sg}(\nabla V_{k+1}(x,y)) = \mathcal{S}$ for all (x,y). A similar argument holds when $\mathrm{sg}(\nabla V_1(x,y)) = -$. ∎

Corollary 3 *For all values of $x, p, w,$ and $k < N$,*

$$u_k^* = \underset{0 \leq u \leq \min(x, M-y)}{\operatorname{argmin}} E_w V_k(x+w-u, y+u), \tag{6.11}$$

is given by

$$u_k^* = \begin{cases} 0 & \text{if} \quad \mathcal{S} = + \quad \forall (x,y), \\ \min(x, M-y) & \text{if} \quad \mathcal{S} = - \quad \forall (x,y). \end{cases} \tag{6.12}$$

Proof. The proof follows from Lemma 6.4.1 and Lemma 6.4.2. ∎

Lemma 6.4.3 *The value function $V_k(x,y)$ is non-decreasing in y for a fixed x and non-decreasing in x for a fixed y.*

Proof. From the assumption, $h(x)$ is increasing in x and $g(y)$ is increasing in y and observe that $V_1(x,y) = h(x) + g(y)$. The proof for a general value of k follows by using induction approach as in the previous proof. ∎

Corollary 4 *For $\mathcal{S} = +$, it is optimal for the moving server to be at the first queue only when the fixed server is activated.*

Proof. For $\mathcal{S} = +$, using Lemma 6.4.1 and Lemma 6.4.2, Equation 6.6 simplifies to

$$V_{k+1}(x,y) = h(x) + g(y) + \min \quad \{E_w V_k((x-(M-y))^+ + w, 0) + c, \tag{6.13}$$

$$E_w V_k(x+w, y), E_w V_k(x+w, (y-\zeta)^+)\}. \tag{6.14}$$

From Lemma 6.4.3, $E_w V_k(x+w,y) \geq E_w V_k(x+w,(y-\zeta)^+)$. Therefore, if $\phi_k = 0$, the moving server serves the second queue. The moving server serves the first queue only when $\phi_k = 1$. ■

Using Lemmas 6.4.1, 6.4.2 and 6.4.3, we can characterize the set of optimal policies based $\mathcal{S}$ as follows. By the set of optimal policies we mean the set of policies which will be optimal for some value of (x,y).

Theorem 6.4.1 *For any $k \leq N$, x and y, the set of optimal policies (i.e., 3-tuples (ϕ^*, u^*, v^*))*

1. *for $\mathcal{S} = +$ is:*

$$\mathcal{A}_+^* = \begin{cases} (1, \min(x, M-y), y + \min(x, M-y)), \\ (0, 0, \min(y, \zeta); \end{cases}$$

2. *for $\mathcal{S} = -$ is:*

$$\mathcal{A}_-^* = \begin{cases} (1, \min(x, M-y), y + \min(x, M-y)), \\ (0, 0, \min(y, \zeta)), \\ (0, \min(x, M-y), 0). \end{cases}$$

The above result is important because now the action set has reduced considerably. From Theorem 6.4.1, the problem can be studied in two different regimes.

1. In the first regime, where $\mathcal{S} = +$, there controller has to decide between two actions. In the first action, the fixed server serves $y + \min(x, M-y)$ from the second queue and the mobile server serves $\min(x, M-y)$ from the first queue. In the second action, the fixed server is inactive and the mobile server serves $\min(y, \zeta)$ from the second queue. Therefore, the controller needs to decide whether to activate the fixed server or not.

2. In the second regime, where $\mathcal{S} = -$, the controller has to decide among three actions. In the first action, the fixed server serves $y + \min(x, M-y)$ from the second queue and the mobile server serves $\min(x, M-y)$ from the first queue. In the second action, the fixed server is inactive and the mobile server serves $\min(y, \zeta)$ from the second queue. In the third action, the fixed server is inactive and the mobile server serves $\min(x, M-y)$ from the first queue. For $\zeta = 0$, the number of decisions reduce to two. In this case, as for $\mathcal{S} = +$, the controller needs to decide whether to activate the fixed server or not.

In order to obtain more structural results, we shall study the two regimes separately in Section 6.5 and in Section 6.6.

We end this section with a property of the value function which will be of use later.

Lemma 6.4.4 *The function* $\frac{\partial}{\partial x}V_k(x,y)$ *(resp.,* $\frac{\partial}{\partial y}V_k(x,y)$*) is bounded below by* $\min_x h'(x)$ *(resp.,* $\min_y g'(y)$*) for each* $k \leq N$.

Proof. We shall use induction to prove this lemma. Suppose $\mathcal{S} = +$ (It can be similarly shown for $\mathcal{S} = -$.). For $k = 1$, $V_1(x,y) = h(x) + g(y)$. The statement is therefore true for $k = 1$. The dynamic programming equation is

$$V_{k+1}(x,y) = h(x) + g(y) + \min\left\{E_w V_k((x-(M-y))^+ + w, 0), E_w V_k(x+w, (y-\zeta)^+)\right\}. \quad (6.15)$$

For a fixed value of w, from Lemma 6.4.3, $\frac{\partial}{\partial x}V_k((x-(M-y))^+ + w, 0)$ and $\frac{\partial}{\partial y}V_k(x+w,(y-\zeta)^+)$ are nonnegative. The same property therefore holds for the convex combination over w weighted by the distribution of w. Therefore, we obtain

$$\frac{\partial}{\partial x}V_{k+1}(x,p) \geq h'(x) \geq \min_x h'(x). \quad (6.16)$$

Using a similar argument, it follows that $\frac{\partial}{\partial y}V_k(x,y) \geq \min_y g'(y)$. ■

6.5 The case of $\mathcal{S} = +$

In this section we study the case $\mathcal{S} = +$ and obtain structural results for the finite/infinite horizon discounted cost problem and then use the *vanishing discount* approach to study the average cost problem.

6.5.1 Structure of an optimal policy

In the previous section it was observed that it was optimal to serve in the first queue only when the fixed server was active. We now characterize the policy for activating the fixed server. We will show that there exists a threshold policy for activating the fixed server. Specifically, for a given value of k and y, there is a threshold value, $x_k^*(y)$, such that if $x < x_k^*(y)$ then the optimal action is to disable the fixed server. We also show that $x_k^*(y)$ is a decreasing function of y.

From Theorem 6.4.1, the controller has two actions to choose from. The dynamic programming equation becomes

$$\begin{aligned} V_{k+1}(x,y) \quad = h(x) + g(y) + \quad & \beta \min\left\{E_w V_k((x-(M-y))^+ + w, 0) + c,\right. \\ & \left. E_w V_k(x+w, (y-\zeta)^+)\right\}. \end{aligned} \quad (6.17)$$

For a given value of k and y, let the first crossover point, $x_k^*(y)$, of $E_w V_k((x-(M-y))^+ + w, 0) + c$ and $E_w V_k(x+w,(y-\zeta)^+)$. be defined as

$$x_k^*(y) \stackrel{\text{def}}{=} \inf\{x : E_w V_k((x-(M-y))^+ + w, 0) + c \geq E_w V_k(x+w,(y-\zeta)^+)\}. \quad (6.18)$$

For a given value of k and y, the threshold $x_k^*(y)$ is that value of x below which it is optimal to disable the fixed server. Such a value always exists since $x_k^*(y) = 0$ is also a possibility.

Theorem 6.5.1 *If* $\mathcal{S} = +$*, then* $x_k^*(y)$ *is a decreasing function of* y *for any fixed* k.

Proof. First, we show that the assertion is true for each fixed value of w. To simplify the notation, let $d_s(x,y) \stackrel{\text{def}}{=} V_k((x-(M-y))^+ + w, 0) + c$, and $d_i(x,y) \stackrel{\text{def}}{=} V_k(x + w, (y-\zeta)^+)$. For a fixed value of k, y, and w, let x^* be the first crossover of $d_s(x,y)$ and $d_i(x,y)$, i.e., it is optimal to remain idle for $x < x^*$. Since x^* is a crossover point,

$$\frac{\partial}{\partial x} d_s(x,y)|_{x=x^*} < \frac{\partial}{\partial x} d_i(x,y)|_{x=x^*}, \tag{6.19}$$

and

$$d_s(x^*, y) = d_i(x^*, y). \tag{6.20}$$

In order to show that $x_k^*(y,w)$ is decreasing in y, first we need to show that $d_s(x^*, y+\delta) < d_i(x^*, y+\delta)$ for some $\delta > 0$. Consider

$$d_s(x^*, y+\delta) - d_i(x^*, y+\delta) = d_s(x^*, y) + \delta \frac{\partial}{\partial x} d_s(x,y)|_{x=x^*} + o(\delta) - d_i(x^*, y) - \delta \frac{\partial}{\partial y} d_i(x^*, y) + o(\delta). \tag{6.21}$$

Substituting from Equations 6.19 and Equation (6.20) in the above equation, we obtain

$$\begin{aligned} d_s(x^*, y+\delta) - d_i(x^*, y+\delta) \;&<\; \delta \frac{\partial}{\partial x} d_i(x,y)|_{x=x^*} - \delta \frac{\partial}{\partial y} d_i(x^*, y) + o(\delta) \\ &= \delta \left[\frac{\partial}{\partial x} d_i(x,y)|_{x=x^*} - \frac{\partial}{\partial y} d_i(x^*, y) \right]. \end{aligned} \tag{6.22}$$

Since $\mathcal{S} = +$, on the right hand side the coefficient of δ is negative for sufficiently small δ. Therefore, $d_s(x^*, y+\delta) < d_i(x^*, y+\delta)$ for some $\delta > 0$. ■

We note here that to obtain the parametric monotonicity we have not used convexity or decreasing differences property of the value function (which are in fact not true in our case).

6.5.2 The infinite horizon discounted cost problem for $y = 0$

Next, we consider the case of $N = \infty$, i.e., the infinite horizon discounted cost problem. The properties obtained for the finite horizon problem hold also for the infinite horizon discounted cost problem. For the finite horizon case, it was necessary to study the value function for all possible values of its argument, i.e., x and y. However, since we intend to study the average cost problem via the infinite horizon discounted cost case by using the *vanishing discount* approach, we shall see that, for the case where $\mathcal{S} = +$, it is sufficient to study the value function $V(x,y)$ only at $y = 0$. Therefore, in this section we study the structure of the infinite horizon discounted problem value function $V(x,0)$ assuming $\mathcal{S} = +$. We start with considering the problem of minimizing the discounted cost

$$V(x_0, y_0) = \sum_{k=0}^{\infty} \beta^k (h(x_k) + g(y_k) + c \cdot \mathbf{I}(\phi_k = 1)).$$

The decision to be made at each slot is whether to activate the fixed server or not. Using the properties of the value function obtained in the finite horizon case, $V(\cdot,\cdot)$, satisfies the dynamic programming equation

$$V(x,y) = h(x) + g(y) + \beta \min \left\{ E_w V((x-(M-y))^+ + w, 0) + c, E_w V(x+w, (y-\zeta)^+) \right\}. \tag{6.23}$$

The value function, $V(x,y)$, can also be obtained as $V(x,y) = \lim_{k\to\infty} V_k(x,y)$. From the value iteration equation Equation (6.23) and the state equations Equation (6.1), we observe that once the second queue becomes empty, it remains empty. The work is queued only at the first queue. The decision to be made is whether or not to activate the fixed server. The value iteration for $y = 0$ is

$$V(x,0) = h(x) + g(0) + \beta \min\left\{E_w V((x-M)^+ + w, 0) + c, E_w V(x+w,0)\right\}. \tag{6.24}$$

The value function $V(x,0)$ depends only on the value function at $y = 0$. Assuming the initial state y_0 to be zero, the second queue always remains empty. This simplifies the problem as we can study the value function as function of x only. The problem now becomes equivalent to optimal control of a discrete-time queue with one server: A maximum of M units can be served in each slot. There is a holding cost of $h(x) + g(0)$ associated with a workload of x. In every slot in which work is done, there is an additional cost of c.

Remark 6.5.1 *We briefly return to the wireless problem and give the analogous policy. In the finite horizon case, the structural results for $\mathcal{S} = +$ imply that either the device is in sleep mode, in which case it recovers some charge, or maximum possible work is done at the transmit buffer and a new battery is reordered. In the infinite horizon case, some further results can be obtained due to the fact that once the battery charge recovers to its maximum capacity, M, it cannot recover further charge. Since the optimal policy is to either operate the device in sleep mode or to serve maximum possible from the transmit queue and reorder the battery, the residual charge remains at M. Hence, we can study the optimal policy by studying the value function at $p = M$ or, equivalently, at $y = 0$.*

In the rest of this section we shall study the structure of the optimal policy for the equivalent single server problem. Since y is equal to zero, we shall use $V(x)$ to denote $V(x,0)$.

Condition B *The function $h(x)$ is a concave and increasing function, and $h'(0) \le \frac{(1-\beta)c}{M}$.*

Theorem 6.5.2 *Under Condition B, the value function $V_k(x)$ is concave in x, and $V_k'(0) \le \frac{(1-\beta^k)c}{M}$ for all $k \ge 0$.*

Proof. We show this using induction. The statement is true for $k = 1$ by virtue of the assumption on $h(x)$. Since $V_k(x)$ is concave, $V_k'(x) \le V_k'(0)$. Therefore, if $V_k(x)$ satisfies the two assertions then for any x and w,

$$V_k(x+w) \le V_k((x-M)^+ + w) + c.$$

Hence, $V_{k+1}(x) = h(x) + g(0) + \beta E_w V_k(x+w)$. Since $V_k(x)$ is concave, it follows that $V_{k+1}(x)$ is also concave. Also,

$$V_{k+1}'(0) = h'(0) + \beta E_w V_k'(w) \le \frac{(1-\beta)c}{M} + \beta\frac{(1-\beta^k)c}{M} = \frac{(1-\beta^{k+1})c}{M}.$$

The assertion is therefore true. ∎

The above result suggests that if $\sup_x h'(x) = h'(0) \le \frac{(1-\beta)c}{M}$ then it is optimal to remain idle *forever* after the queue becomes empty. This result, though seemingly counter-intuitive, can be explained by the fact that we are considering discounted cost which gives more weight to the near future cost. To be able to give more consideration to the distant future, we require β very close to

unity, in which case the condition of the theorem (i.e., $\sup_x h'(x) = h'(0) \leq \frac{(1-\beta)c}{M}$) will not hold. This point will be clear when we consider the average cost optimization problem where one gives all weight to the distant future costs.

Corollary 5 *Under Condition B, the infinite horizon discounted cost problem value function has a slope* $V'(x) \leq \frac{c}{M}$.

For the linear cost function $h(x)$, we have the following corollary.

Corollary 6 *If* $h(x) = c_1 x$ *with* $c_1 \leq \frac{(1-\beta)c}{M}$ *then*

1. $V_k'(0) = \frac{1-\beta^k}{1-\beta} c_1$, *and*
2. $$V(x) \stackrel{\text{def}}{=} \lim_{k\to\infty} V_k(x) = \frac{c_1 x + g(0)}{1-\beta} + c_1 E[w] \frac{\beta}{(1-\beta)^2}. \tag{6.25}$$

These results hold when $h(x)$ is a concave function, and $h'(0) \leq \frac{(1-\beta)c}{M}$ (i.e., $\sup_x h'(x) \leq \frac{(1-\beta)c}{M}$.). Under these conditions, the optimal policy is to always remain idle. Next, we find a condition under which the optimal policy will require some work to be done.

Condition C *The function* $h(x)$ *is a concave and increasing function, and* $\inf_x h'(x) > \frac{(1-\beta)c}{M}$.

Theorem 6.5.3 *Under Condition C there exists an integer* K *such that* $V_K(x) > V_K(0) + c$ *for some* $x < M$.

Proof. We prove this by contradiction. For a given $x < M$, suppose there exists no such K, i.e., $V_k(x+w) < V_k(w) + c$, $\forall k$. We start from $k = 2$ and show that $V'(x)$ is increasing. Due to the assumption on $E_w V_k(x+w)$, we have

$$V_2(x) = h(x) + g(0) + \beta E_w V_1(x+w). \tag{6.26}$$

Since $V_1(x) = h(x) + g(0)$ is concave in x, it follows that $V_2(x)$ is concave in x, and by assumption, satisfies the property that $V_2(x+w) < V_2(w) + c$ for each $x < M$. Therefore, concavity in x is preserved for all k. Also,

$$V_2'(x) \geq \inf_x h'(x) + \beta \inf_x h'(x) = \frac{(1-\beta^2)c}{M}. \tag{6.27}$$

Proceeding in similar manner, we can show that for each fixed x, $V_k'(x)$ is strictly increasing in k and is in fact at least $\frac{1-\beta^k}{1-\beta} \inf_x h'(x) > (1-\beta^k)\frac{c}{M}$ as can be seen from the above. This contradicts the assumption that for each k, $V_k(x+w, M) < V_k(w, M) + c$ for each $x < M$. ∎

Until now $h(x)$ was assumed to be a concave function. Next, we consider another possible structure of $h(x)$.

Definition 1 *A function* $f(x)$ *is said to be* M*-convex if* $f'(x) \leq f'(x+M)$ *for all* x.

Theorem 6.5.4 *If $h(x)$ is M-convex and increasing then $V_k(x)$ is continuous and M-convex, i.e.,*

$$V_k'(x+M) \geq V_k'(x), \quad x \geq 0. \tag{6.28}$$

Proof. We show this by induction. The assertion is true for $k = 1$. Assume above is true for $k \geq 1$. From the dynamic programming equation

$$\begin{aligned} V_{k+1}'(x+M) - V_{k+1}'(x) = h'(x+M) - h'(x) \\ + \frac{\partial}{\partial x} \min\left\{E_w V_k((x+w)+c, E_w V_k(x+M+w)\right\} \\ - \frac{\partial}{\partial x} \min\left\{E_w V_k((x-M)^+ + w) + c, E_w V_k(x+w)\right\} \end{aligned} \tag{6.29}$$

The required equality can be shown by comparing each term under the first minimum operator with each term under the second minimum operator, and using M-convexity of $V_k(x)$ and $h(x)$. ■

The above result also implies that $V_k(x)$ need not be convex or concave in general.

Let us assume that $\inf_x h'(x) > 0$. We now scale the parameters of the function g and c such that $\inf_x h'(x) = 1$. We shall now use Lemma 6.4.4 to establish the optimality of the threshold based policy.

Theorem 6.5.5 *If $h(x)$ is M-convex and increasing with $h'(0) = 1$ and $c < M$ then for the infinite horizon problem with the second queue empty, there is a unique threshold η^* such that if $x \leq \eta^*$ then it is optimal for the fixed server to be inactive in the present slot else it is optimal to activate the fixed server in which case the mobile server serves* $\min(x, M)$ *from the first queue in the present slot.*

Proof. Since we are considering the infinite horizon problem, the value function $V(x)$ (with the second queue empty) satisfies Equation 6.23 with $y = 0$. First, we show that there exists at least one point at which it is optimal to serve $\min(x, M)$. From Lemma 6.4.4 and the assumption that $\inf_x h'(x) = 1$, the derivative of $V(x)$ is bounded below by unity. As a consequence of this, $V(M+x) \geq V(x) + M$ for any x. Let x be equal to M. We compare the two terms under the minimum operator in Equation 6.24. For $x = M$,

$$\begin{aligned} E_w V(w) + c - E_w V(M+w) &= c + \int_w (V(w) - V(M+w)) d\mu(w) \\ &\leq c - M. \end{aligned} \tag{6.30}$$

Since $c < M$, it follows from above that

$$c + E_w V((M-M)^+ + w) < E_w V(M+w)$$

so that the optimal choice in a slot in which $x = M$ is to serve $\min(x, M)$.
Also, at $x = 0$,

$$E_w V((x-M)^+ + w) + c - E_w V(x+w) = c > 0.$$

Therefore, it is optimal to leave the server idle in a slot in which $x = 0$.

Next, we show that if it is optimal to serve $\min(x, M)$ at x then it is optimal to serve $\min(z, M)$ at any $z > x$. Let $x \geq M$ be a point such that $c + E_w V((x-M)^+ + w) < E_w V(x+w)$. We know that $x = M$ is such a point. For any $\delta > 0$,

$$\begin{aligned}
& c + \int_w V((x+\delta-M)+w)\mu(dw) \\
&= c + \int_w \big[V((x-M)+w) + \delta V'((x-M)+w) + o(\delta)\big]\mu(dw) \\
&< \int_w \big[V(x+w) + \delta V'((x-M)+w) + o(\delta)\big]\mu(dw) \\
&< \int_w \big[V(x+w) + \delta V'(x+w) + o(\delta)\big]\mu(dw) \\
&= \int_w V(x+\delta+w)\mu(dw) + o(\delta),
\end{aligned}$$

where the last inequality follows from Theorem 6.5.4. Thus, if it is optimal to activate the fixed server and serve $\min(x, M)$ from the first queue when the occupancy of the first queue is x, then the same is also optimal for all $(x + \delta) > x$, and in particular, for all $x \geq M$.

For $x < M$, we note that $E_w V((x-M)^+ + w) + c$ is a constant independent of x, and $E_w V(x+w)$ is an increasing function of x. Also, at $x = 0$ it is optimal to disable the fixed server whereas at $x = M$ it is optimal to activate the fixed server and serve $\min(x, M)$ from the first queue. Therefore, there exists a $\eta^* < M$ such that $E_w V(w) + c = E_w V(\eta^* + w)$. For $\eta^* < x < M$, $E_w V(w) + c$ is less than $E_w V(\eta^* + w)$, and for $0 \leq x < \eta^*$, $E_w V(w) + c$ is greater than $E_w V(\eta^* + w)$. Since it is also optimal to serve $\min(x, M)$ for $x \geq M$, it is therefore optimal to disable the fixed server for $x < \eta^*$ and activate the fixed server and to serve $\min(x, M)$ from the first queue for $x \geq \eta^*$. This completes the proof of the existence of a threshold type policy. ■

We note that linear functions are both concave and M-convex. For a linear cost function $h(x) = c_1 x$, we can use Theorems 6.5.2, 6.5.3, and 6.5.5 to obtain the following optimal policy when starting from $y_0 = 0$. For a discount factor of β,

1. if $c_1 \leq (1-\beta)\frac{c}{M}$ then it is optimal to always remain idle.

2. else there is a $\eta^* < M$ such that it is optimal to remain idle for $x < \eta^*$ and serve $\min(x, M)$ for $x > \eta^*$.

The first part is obtained from Theorem 6.5.2. The second part is obtained as follows. Since $c_1 > (1-\beta)\frac{c}{M}$ and $y_0 = 0$, using Theorem 6.5.3, we will ultimately get to a slot, say K, in which $V_K(x) > V_K(0) + c$ for some $x < M$. Since the structure derived for M-convex $h(x)$ is valid here, Theorem 6.5.5 can be invoked.

We now characterize the effect of the discount factor on the value function. Let $V_{k,\beta}(\cdot,\cdot)$ and $V_\beta(x)$ denote respectively the k-step to go value function and the limiting value function when the discount factor is β.

Lemma 6.5.1 *For each k and x, $V_{k,\beta}(x)$ is non-decreasing in β.*

Proof. We show this by induction. The result is true for $k = 1$ because $V_{1,\beta}(x) = h(x) + g(0)$. Assume now that the assertion is true for some k. From Equation (6.23),

$$\frac{d}{d\beta}V_{k+1,\beta}(x) = \min\left\{E_w V_{k,\beta}((x-M)^+ + w) + c, E_w V_{k,\beta}(x+w)\right\} + \beta\frac{d}{d\beta}\min\left\{E_w V_{k,\beta}((x-M)^+ + w) + c, E_w V_{k,\beta}(x+w)\right\}.$$

Each term of the right hand side is positive. Therefore, $V_{k+1,\beta}$ is also non-decreasing in β. ■

Lemma 6.5.2 *The function $\frac{\partial}{\partial x}V_{k,\beta}(x)$ is a non-decreasing function of β.*

Proof. The statement is true for $k = 1$. It can be shown using induction that the assertion holds for all $k \geq 1$. ■

Let $\eta^*(\beta)$ denote the unique threshold of the infinite horizon discounted cost problem when the discount factor is β.

Theorem 6.5.6 *The optimal threshold $\eta^*(\beta)$ is non-increasing in β.*

Proof. Since $\eta^* < M$ is the unique threshold,

$$E_w V_\beta(\eta^*(\beta) + w) - E_w V_\beta(w) = c. \tag{6.31}$$

For an $\epsilon > 0$ such that $\beta + \epsilon < 1$, using Lemma 6.5.2, it follows that $E_w V_{k,\beta+\epsilon}(\eta^*(\beta) + w) - E_w V_{k,\beta+\epsilon}(w) > c$ because in between $x = 0$ and $x = \eta^*(\beta)$, the slope of $E_w V_{k,\beta+\epsilon}(x + w)$ with respect to x is at least that of $E_w V_{k,\beta}(x + w)$. This implies that $x_k(\beta + \epsilon) \leq x_k(\beta)$. ■

6.5.3 The average cost problem

We now consider the problem of optimization of the infinite horizon average cost for the case $\mathcal{S} = +$. The approach is to use results from infinite horizon discounted cost optimization and then use the standard vanishing discount approach with $\beta \to 1$. If the average cost exists, it is independent of the initial state so that we can without loss of generality assume that $y_0 = 0$. For the infinite horizon discounted cost case, it was noted that once the second queue became empty it remained so thereafter. Thus making the structural results obtained in Section 6.5.2 for this particular case very relevant to the analysis of average cost problem.

A sufficient condition for existence of a stationary average optimal policy, which can be obtained as limit of discounted cost optimal policies $f_\beta(x)$ is provided in [Sch93]. In our problem $f_\beta : \mathcal{R} \to \{0, 1\}$ where 0 means the fixed server is disabled, and 1 means the fixed server is activated and the mobile server serves $\min(x, M)$ from the first queue. Define $w_\beta(x) = V_\beta(x) - \inf_x V_\beta(x)$, where $V_\beta(x)$ is the infinite horizon discounted cost value function when discount factor is β, the first queue's occupancy is x and the second queue is empty. The reference [Sch93] gives a set of conditions on the problem definition (which hold in the present case but we do not provide their details here) under which the following result holds:

Theorem 6.5.7 ([Sch93] Theorem 3.8) *Suppose there exists a policy Ψ and an initial state x such that the average cost $V^{\Psi}(x) < \infty$. Let $\sup_{\beta<1} w_\beta(x) < \infty$ for all x, then there exists a stationary policy f_1 which is average cost optimal and the optimal cost is independent of the initial state. Also f_1 is limit discount optimal in the sense that, for any x and given any sequence of discount factors converging to one, there exists a subsequence $\{\beta_m\}$ of discount factors and a sequence $x_m \to x$ such that $f_1(x) = lim_{m\to\infty} f_{\beta_m}(x_m)$.*

In order to apply above result we need to show:

1. Existence of policy Ψ: Since $E[w] < M$, the policy of serving $\min(x, M)$ in every slot yields a finite average cost.

2. $\sup_{\beta<1} w_\beta(x) < \infty$ for all x: For any β, since $V_\beta(x)$ is monotone increasing, it follows that $x^* := \underset{x}{\operatorname{argmin}}\, V_\beta(x) = 0$. We have

$$V_\beta(x^*) = h(0) + \beta E_w V_\beta(w) \tag{6.32}$$

Now, for any fixed $x_0 = x$, consider a policy that serves from the first and second queue till the time the first queue is empty (let us denote this time by a random variable Z). Then, it can be seen that

$$\begin{aligned} V_\beta(x) &\leq \sum_{n=\lceil \frac{x}{M} \rceil}^{\infty} \left[\sum_{j=0}^{n} \beta^j h(x + \sum_{i=1}^{j} w_i - (j+1)M) + \beta^n V_\beta(0) \right] \text{Prob}(Z = n) \quad (6.33) \\ &\leq \sum_{n=\lceil \frac{x}{M} \rceil}^{\infty} \left[\sum_{j=0}^{n} \beta^j h(x + \sum_{i=1}^{j} w_i - (j+1)M) \right] \text{Prob}(Z = n) + V_\beta(0). \quad (6.34) \end{aligned}$$

Therefore,

$$w_\beta(x) = V_\beta(x) - V_\beta(0) \leq \sum_{n=\lceil \frac{x}{M} \rceil}^{\infty} \left[\sum_{j=0}^{n} h(x + \sum_{i=1}^{j} w_i - (j+1)M) \right] \text{Prob}(Z = n). \tag{6.35}$$

the expression on the right hand side is independent of β and finite almost surely if $E[W] < M$. The required condition is thus verified.

Hence, for the average criterion, the cost is independent of the initial state. Thus, we can, without loss of generality, assume that the second queue is initially empty. We can now make use of the results of Section 6.5.2 and obtain our main result.

Theorem 6.5.8 *For the average cost optimization problem, if $\mathcal{S} = +$ and $h(x)$ is a M-convex function then there exists a threshold based policy which gives the minimum cost.*

Proof. We have seen that for the discounted cost case the optimal policy is threshold based. We also know that the threshold is a monotone function of the discount factor β. This, along with Theorem 6.5.7, implies the existence of a threshold based optimal policy for average cost case as well. ■

6.5.4 Numerical results

We now illustrate the obtained analytical results with numerical examples. We note that the experiments are performed using a discrete state space. The holding costs are assumed to be linear in the queue lengths, i.e., $h(x) = c_1 x$, and $g(y) = c_2 y$. The number of arrivals in a slot is geometrically distributed with mean $E[w] = 10$. The parameters of the numerical evaluation are set as follows: $c_1 = 1$, $c_2 = 2$, $c = 30$, $M = 20$, $\beta = 0.97$. We note that $\mathcal{S} = \text{sg}(-h'(x) + g'(y))$ is positive. Also, $c_1 \geq \frac{(1-\beta)c}{M}$.

Remark 6.5.2 *We have taken ζ to be zero for these experiments. However, the structure that we obtained for $\mathcal{S} = +$ holds for $\zeta \geq 0$.*

For numerical computation the length of the second queue is taken to be 400 which is much larger than $E[w]$ and M. The initial state of the second queue is taken to be zero. The results are shown for the infinite horizon discounted cost problem.

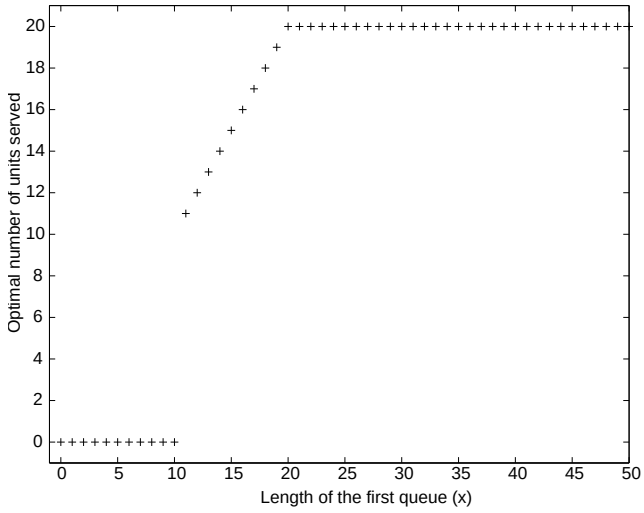


Figure 6.3: The optimal number of units served versus the first queue length, x.

In Figure 6.3, the optimal policy is shown as a function of the first queue length. As it was noted earlier, there exists a threshold $\eta^* < M$ such that if $x \geq \eta^*$ then it is optimal to serve $\min(x, M)$ else it is optimal to remain idle. The corresponding value function is shown in Figure 6.4. Even though the cost functions are linear, the value function is neither convex nor concave.

6.6 The case of $\mathcal{S} = -$ and $\zeta = 0$

For the case $\mathcal{S} = -$ we make the simplification that $\zeta = 0$. This will allow us to obtain structural results for the optimal policy.

In Theorem 6.4.1 it was observed that it was optimal for the mobile server to serve $\min(x, M - y)$ from the first queue. We shall characterize the service policy for the fixed server. We will use the same methodology as in the previous section. First, we will solve the finite horizon problem. Then, we will solve the infinite horizon discounted problem, and finally we will solve the average cost problem.

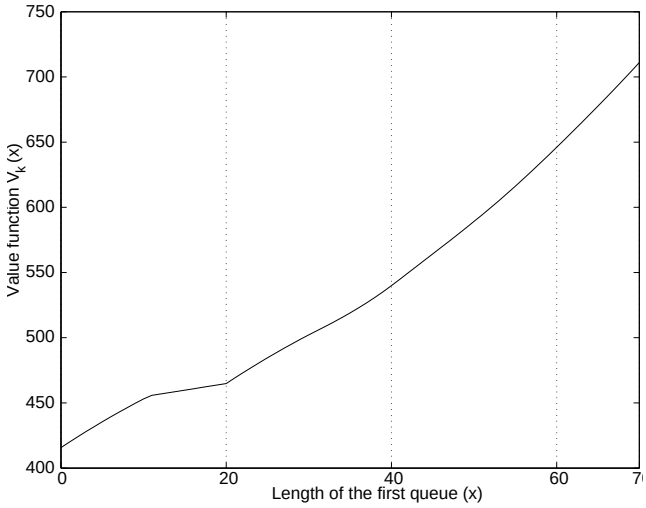


Figure 6.4: The optimal value versus the length of the first queue, x.

The dynamic programming equation for this case is

$$V_{k+1}(x,y) = h(x) + g(y) + \beta \min\{E_w V_k((x-(M-y))^+ + w, 0) + c, \\ E_w V_k((x-(M-y))^+ + w, y + \min(x, M-y)\}. \tag{6.36}$$

The terms under the minimization operator are functions of $(x+y)$. Therefore, the optimal policy depends only on $(x+y)$, i.e., the optimal policy is the same for all (x,y) such that $x+y =$ constant. This structure helps us to characterize the optimal policy.

6.6.1 The finite horizon discounted cost problem

Since the optimal policy is the same for all (x,y) such that $x+y =$ constant, we shall study the optimal policy in two different regions based on $(x+y)$. First, we will study the case $x+y < M$. Later, we shall study the case $x+y \geq M$.

For $x+y < M$, Equation (6.36) becomes

$$V_{k+1}(x,y) = h(x) + g(y) + \beta \min\{E_w V_k(w,0) + c, E_w V_k(w, x+y)\}. \tag{6.37}$$

Let $h(x)$ be a positive and increasing function.

Theorem 6.6.1 *For a given k and $x+y < M$, there exists a threshold η_k^* such that if $\eta_k^* \leq x+y < M$ then it is optimal to activate the fixed server else if $0 \leq x+y < \eta_k^*$ it is optimal to disable the fixed server.*

Proof. We again note that the threshold depends uniquely on $(x+y)$ and is the same for all (x,y) such that $x+y =$ constant. We vary $x+y$ from zero to M. The first term under the minimization operator is independent of x and y and is therefore a constant. From Lemma 6.4.3, $V_k(\cdot,\cdot)$ is increasing in its second argument and therefore the second term under the minimization operator is increasing in $x+y$. We note that, for $(x,y) = (0,0)$, the optimal policy is to remain

idle. As $x+y$ increases, $E_w V_k(w, x+y)$ also increases. Let η_k^* be the point such that $E_w V_k(w, \eta_k^*) = E_w V_k(w, 0) + c$. It is possible that, for $x + y < M$, this point may not exist in which case we set $\eta_k^* = M$. ■

Next, we derive the structure for $x + y \geq M$. For $x + y \geq M$, Equation 6.36 becomes

$$V_{k+1}(x, y) = h(x) + g(y) + \beta \min\{E_w V_k((x + y - M) + w, 0) + c, \\ E_w V_k((x + y - M) + w, M)\}. \tag{6.38}$$

Theorem 6.6.2 *For $x + y \geq M$, if $\inf_y g(y) \geq \frac{c}{M}$ then it is always optimal to activate the fixed server at (x, y) for any k.*

Proof. We note that the optimal policy again depends only on $(x + y)$. From Lemma 6.4.4, if $\sup_y g(y) \geq \frac{c}{M}$ then $\frac{\partial}{\partial y} V_k(x, y) \geq \frac{c}{M}$ for all values of k and (x, y). Therefore, for any given value of $x + y \geq M$, $E_w V_k((x + y - M) + w, 0) + c \leq E_w V_k((x + y - M) + w, M)$. Hence, it is optimal to activate the fixed server for all $x + y \geq M$. ■

The condition $\inf_y g(y) \geq \frac{c}{M}$ also implies that $\eta_k^* < M$ for any k. For $\inf_y g(y) \geq \frac{c}{M}$, we can combine the results of Theorems 6.6.1 and 6.6.2 to obtain a threshold type policy for the fixed server.

Theorem 6.6.3 *For each k, if $\inf_y g'(y) \geq \frac{c}{M}$ then there exist a $\eta_k^* \leq M$ such that if $x + y \geq \eta_k^*$ then it is optimal to activate the fixed server else it is optimal to disable the fixed server.*

The results obtained here are very similar to those obtained for the case $\mathcal{S} = +$ in the sense that we get a threshold based policy where existence of a nontrivial threshold depends on the slope of the cost functions.

6.6.2 The infinite horizon discounted cost and average cost problems

We note that the existence of the threshold based policy was shown for all k. Therefore, a threshold based policy also holds for the infinite horizon discounted cost problem.

For the average cost problem, we need equivalents of Lemma 6.5.2 and Theorem 6.5.6. Let $\eta_k^*(\beta)$ denote the unique threshold for k-step to go cost function when the discount factor is β.

Lemma 6.6.1 *The function $\frac{\partial}{\partial y} V_{k,\beta}(x, y)$ is non-decreasing in β for each x, k and y.*

Proof. The statement is true for $k = 1$. Assume it is true for all $n \leq k$. Then, $\frac{\partial}{\partial y} V_{k+1,\beta}(x, y)$

$$= \begin{cases} g'(y) + \beta \frac{\partial}{\partial y} E_w V_{k,\beta}(w, x + y) & (x + y) < \eta^*(\beta) \\ g'(y) + \beta \frac{\partial}{\partial y} E_w V_{k,\beta}((x + y - M)^+ + w, 0) & (x + y) > \eta^*(\beta) \end{cases}$$

so that the result follows for $k + 1$-step cost as well. ■

Theorem 6.6.4 *The optimal threshold $\eta_k^*(\beta)$ is non-increasing in β.*

Proof. The proof follows the proof of Theorem 6.5.6. ■

We can now approach the infinite horizon discounted cost and the average cost problems in a way similar to that for $\mathcal{S} = +$. In particular, we get counterparts of Theorems 6.5.5 and 6.5.8.

Theorem 6.6.5 *For the infinite horizon discounted cost problem, if $g(y)$ is an increasing function with $\inf_y g'(y) \geq \frac{c}{M}$ then there is a unique threshold $\eta^*(\beta)$ such that if $x \leq \eta^*(\beta)$ then it is optimal to to disable the fixed server else it is optimal to activate the fixed server.*

Theorem 6.6.6 *For the average cost optimization problem, there exists a threshold based policy which gives the minimum cost.*

Remark 6.6.1 *We revisit the wireless problem in order to relate its optimal policy with that of the tandem queue problem. For $\mathcal{S} = -$ and $\zeta = 0$ (i.e., the battery cannot recover charge when left idle), the optimal policy for replacing the battery is of threshold type. In each slot, $\min(x, p)$ packets are transmitted. The battery is replaced only when $x + M - p$ exceeds a threshold which is less that M.*

6.6.3 Numerical results

We now illustrate the obtained analytical results with numerical examples. The holding costs are assumed to be linear in the queue lengths, i.e., $h(x) = c_1 x$, and $g(y) = c_2 y$. The number of arrivals in a slot is geometrically distributed with mean $E[w] = 10$. The parameters of the numerical evaluation are set as follows: $c_1 = 1$, $c_2 = 0.9$, $c = 15$, $M = 20$, $\beta = 0.97$. We note that $\mathcal{S} = \text{sg}(-h'(x) + g'(y))$ is negative. Also, $c_2 \geq \frac{c}{M}$. The results are shown for the infinite horizon discounted cost problem.

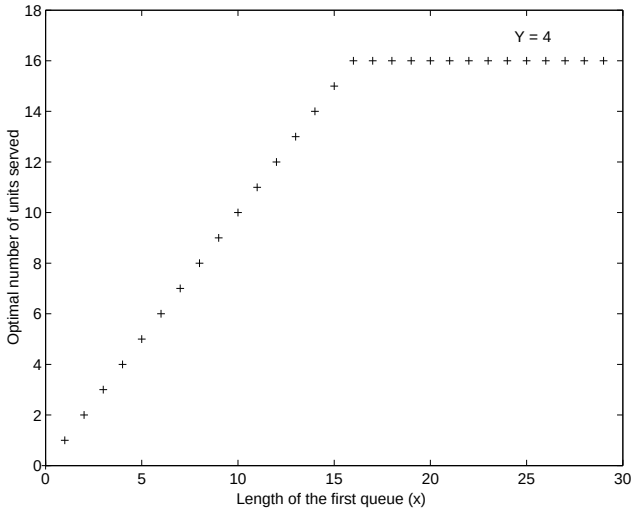


Figure 6.5: The optimal number of units served versus the first queue length, x.

In Figure 6.5, the optimal policy at the first queue is shown as a function of the length of the first queue for $y = 4$. As it was noted in Theorem 6.4.1, u^* is $x \wedge (M - y)$. In Figure 6.6, the optimal policy at the second queue, v, is shown as a function of (x, y). There exists a $\eta^* < M$ such

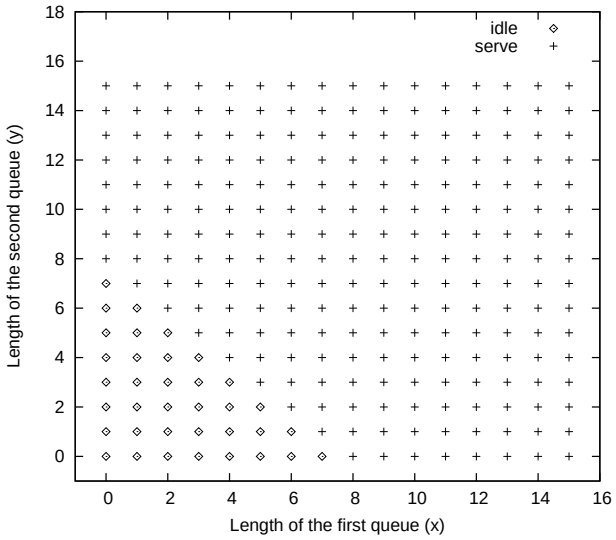


Figure 6.6: The optimal policy at the second queue.

that if $x + y < \eta^*$ then it is optimal to disable the fixed server else it is optimal to activate the fixed server.

6.7 Summary and conclusions

We considered jointly optimal control of delay and energy in a wireless device. An equivalence was established with the control of a tandem queue. The optimal control of tandem queue was formulated as a Markov decision process problem. We considered the optimization of the finite and infinite horizon discounted costs as well as that of infinite horizon average cost. The problem becomes hard as the underlying state space is two-dimensional and second order properties like convexity or increasing/decreasing differences do not hold. We established the optimality of a bang-bang policy which is threshold based. We also studied the behavior of this threshold and obtained parametric monotonicity results. Unlike in other related works which assume certain conditions on the cost functions in order to show the optimality of threshold type policies, we characterize the conditions (e.g., M-convexity) which lead to threshold type policy. That is, our approach to showing optimality of threshold type policy is complementary to the standard approach. Also, we note that the finite horizon model considered in this work is that of a general interacting system where a change in one dimension results in a corresponding change in the other dimension. The use of the sign of directional derivative to establish optimality of bang-bang control for such systems appears to be a natural choice. For finite buffer systems, we developed the notion of M-convexity which ensures the existence of threshold based policies.

Chapter 7

Scheduling of TCP sessions over UMTS

In the final chapter of the thesis we study a problem which is motivated by the transfer of data on the cellular technology, UMTS. Data transfer on the UMTS downlink can be done either through dedicated channels (DCH) or shared channel (FACH). The data transfers on the DCHs can receive very high throughputs. However, a setup time of the order of 250ms is required before data transfer can begin on the DCH. For very short transfers it may be better to use the shared channel. However, for long data sessions it might be better to allocate a dedicated channel. In this paper, we propose a threshold policy to determine which sessions should use the dedicated channels. Initially, we use the FACH for a given connection. Then if we get an indication that the current burst might be long (for example, we observe a long queue from that source), then beyond some threshold we shall try to allocate a DCH to that connection (if there is one available). We also present and evaluate the performance of algorithms in which a timer is set when the queue size falls below a threshold and the connection is on DCH. The use of a timer allows a connection to remain for a longer duration on DCH and thus improve performance. We use *ns-2* simulations to obtain the performance measures for the threshold and timer based policies.

The results in this chapter have appeared in [PAAD03].

7.1 Introduction

UMTS is the 3rd generation of cellular wireless networks which aims to provide high speed data access along with real time voice calls [HT01]. On the UMTS downlink (i.e., from the base station to the mobile), data can be transferred through dedicated channels (DCH) or a shared channel (FACH). Dedicated channels offer higher transfer speeds but require a setup time which is significant (of the order of 250ms). Shared channels, on the other hand, have a low setup time and also low transfer speeds. For sporadic packets (i.e., files or transfers that are very short) it is efficient to use the shared channel. However, for long data sessions it may be better to allocate a dedicated channel. Most data transfers over the Internet are performed today with TCP. Data connections on the Internet are known to be bursty where the basic burst size is best described with a Pareto distribution:

$$\text{Prob}(\text{burst size} > s] = (k/s)^{\beta}, \tag{7.1}$$

where $k > 0$ is the minimum size and $1 < \beta \leq 2$ (for Internet traffic) [Mah97]. We use the traffic description of Appendix B of [UMT98] which is the standard for UMTS data structure. It is also based on the Pareto distribution for the transfer size. Our starting point is the observation that long bursts are in some sense "predictable": if we observe that the current burst has already transfered many packets, say x, then the distribution of the remaining size of the burst $R =$ burst size $- x$ is given by

$$\text{Prob}(R > x + s | R > x) = \left(\frac{x}{x+s}\right)^{\beta}. \tag{7.2}$$

So, for a fixed value of s, as x increases (to infinity), $Pr(R > x + s|R > x)$ increases to 1. (This is a general property of heavy tailed distributions [GK98]). Therefore, it seems natural to implement the following policy: we start by scheduling a new connection to FACH if a DCH is not available. Then, if we get an indication that the current burst might be long (for example, we observe a long queue from that source), then beyond some threshold we shall try to allocate a DCH to that connection (if there is one available). When the queue size of a connection falls below a threshold we switch it back to FACH. We also present and evaluate the performance of a modified threshold algorithm. In the modified threshold algorithm, instead of switching immediately when the queue size falls below a threshold, a timer is set and the connection remains on the DCH for this period. If there are no new arrivals within this timeout period, the connection is switched back to FACH. The timer is used in order to allow the ACKs to reach the TCP source and new packets to be released by the source.

The rest of the chapter is organized as follows. In Section 7.2, we present the network configuration used in the simulations and describe the methods used in the implementation of the code. In Section 7.3, we describe the source traffic model, the simulation parameters and give the values used in the simulations. In Section 7.4, we present the results of the threshold algorithm. In Section 7.5, we describe the modified threshold policies wherein a timer is used along with the threshold on the DCH. In Section 7.6, we present the results of the modified threshold policies and compare them with the results of the original threshold policy. In Section 7.7, we summarize the results obtained from this work and present the conclusions.

7.2 Network model

We consider a network model with N_{tcp} TCP sources which need to send data to mobile receivers. We assume a single cell scenario with one base station and several mobile stations which act as destinations for TCP traffic. The TCP sources are assumed to be connected to the base station of the cell with a high speed (5mbps, 30ms) link. The base station can transfer data from a TCP source on either DCH or FACH at a given time. There is one FACH and N_{dch} DCHs in the system. The FACH is a time division multiplexed channel. In addition to any TCP connections which may be present on a FACH, there is signaling traffic which must be transmitted on the FACH. The signaling traffic has priority over the TCP connections. During the silence periods of the signaling traffic, data from one or more TCP connections can be transmitted on the FACH. On the FACH, data from the TCP connections is assumed to be transmitted in a round-robin fashion. If all the DCHs have a TCP connection configured, a connection on DCH should be first switched to FACH before a session from FACH can be switched on to a particular DCH. This means that a switch can take up to 500ms (if there is already a TCP connection configured on the DCH).

Switching from one channel to another is costly in time and in signaling. In the model we assume that there exists a queue corresponding to each TCP connection in the base station. The

base station is hence able to track the queue length of each connection. During the switching time (of around 250ms) from one channel to another, no packets from the queue of the TCP connection being switched can be transmitted. We, therefore, try to avoid quick switching and propose the following channel allocation policy. There are two thresholds T_h and T_l. If the number of packets in the queue of a connection using FACH reaches T_h and a DCH is available then we initiate a switch to the DCH. If the number of packets in the queue of a connection using DCH drops below T_l and the number of allocated DCHs equals the N_{dch}, then we switch the connection back to FACH. The simulation setup for the network is presented in Figure 7.1.

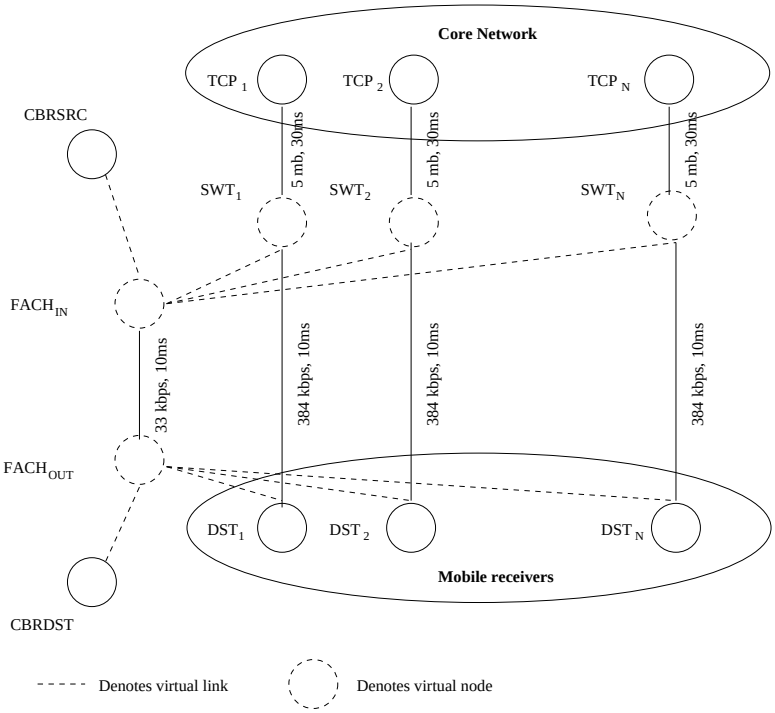


Figure 7.1: Simulation setup

Each TCP source node, TCP_i is connected to a routing node called Switch (SWT_i). SWT_i is present inside the base station and can be connected either to the $FACH_{IN}$ or directly to the TCP destination via the DCH. The SWI_i node is created to simplify the simulations and may not be present inside a real base station. The $FACH_{IN}$ is another virtual node which simulates the prioritized round robin service discipline taking place on the FACH. This is a service discipline which we had to implement. In this discipline, the node $FACH_{IN}$ gives priority to the traffic from CBRSRC while serving the packets from the SWI_i's (only those which are currently not transmitting on DCH) in a round-robin manner. We note that there are no queues at $FACH_{IN}$ and all the packets are either queued at SWT_i or at the $CBRSRC$. The $CBRSRC$ simulates a constant bit rate source of signaling/control traffic. It generates packets at rate R_{sig} and is assumed to be present within the base station. The signaling traffic flows from the base station to the mobile receivers. Even though we model the destination of the signaling traffic ($CBRDST$) as one node different from DST_i, we note that it does not affect the simulations as simultaneous transfer of data and control packets to the same mobile receiver is possible when different channels are used. The links $SWI_i - FACH_{IN}$ and $CBRSRC - FACH_{IN}$ are virtual links within the base station and thus have zero delay. We note that the data from SWT_i to DST_i can take two different routes

i.e., $SWI_i - FACH_{IN} - FACH_{OUT} - DST_i$ (via FACH) or $SWT_i - DST_i$ (via DCH). At any given time only one route from the above two can be active. Although in the simulation scenario we have as many DCH links as TCP source nodes, the simulation allows us to connect not more than N_{dch} DCH channels at a time, which may be chosen strictly smaller than the number of TCP sources (N_{tcp}). In the simulations we switch over from FACH to DCH by changing the cost of the links and recomputing the routes. This is done as follows: Initially, the cost of the direct path from the Switch to the TCP destination is set to 10 and the cost of all other links to 1. Hence, the traffic gets routed through the FACH. When a switch is desired, the cost of the DCH is set to 1 and the routes are recomputed. This activates the DCH and the traffic gets routed on the DCH.

7.3 Input for the simulation

We use *ns-2* [MF] for our simulation study with the following parameters:

- Number of available DCH channels (N_{dch}) is taken to be 1. The number of simultaneous TCP connections, N_{tcp} varies between 2 to 8.
- Values of threshold T_h for switching between channels is varied between 0 and 15. A threshold value of 0 indicates that we try to switch a connection to DCH as soon as it arrives.
- The duration of the simulation is taken 200000 secs in order to reach stationarity. The large time needed to get good accuracy is partially due to the nature of the long range dependence of the traffic.
- The time it takes to switch between channels (D_{sw}) is 250ms.
- We consider the background (non TCP traffic source) traffic source, that uses the FACH, to be CBR source with rate $R_{sig} = 24$ kbps. It sends a 1kB packet at an interval of 1/3secs. It has non pre-emptive priority over TCP.
- The TCP connection traffic model is as follows: On a TCP connection, data arrives in bursts. The number of packets in a burst has a Pareto distribution. The shape parameter is generally taken to be between 1.04 and 1.14 [Mah97, GS99]. We take the shape parameter to be $k = 1.1$ [UMT98] and the average file (burst) size is taken to be $FS_{avg} = 30kB$ [Mah97]. We do not take all the parameters from [UMT98] as they are not adapted to TCP traffic. The model in [UMT98] considers packets of size 81.5 bytes whereas TCP does not send short packets [Bra89]. A TCP connection alternates between "ON" and "OFF" states. In the ON state, the inter-arrival time between successive bursts is exponentially distributed with mean T_{bur}. At the end of each burst, the connection goes into OFF state with probability P_{off}. It remains in the OFF state for an exponentially distributed duration with mean T_{off}. At the end of this duration the connection goes to ON state which is marked by the arrival of a burst. Figure 7.2 shows the traffic model used at each source node.
- The TCP packet size is taken to be 320 bytes which is of the same size as a RLC data segment in UMTS [TSG99].
- The schedule of service of packet in the FACH is taken to be round robin. The implementation in *ns-2* is not standard and we had to change some link element features of *ns-2* in order to implement this schedule.

Tables 7.1 and 7.2 give the values of the network parameters and traffic parameters, respectively.

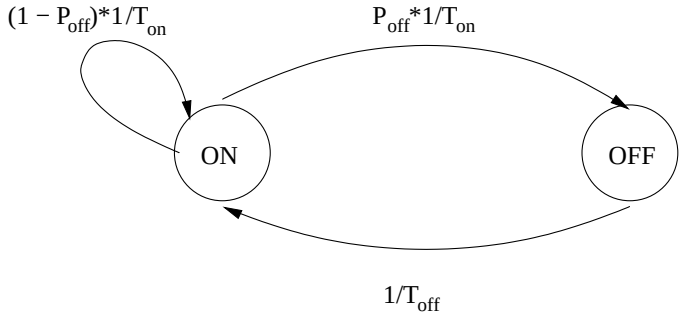


Figure 7.2: Traffic source model

7.4 Simulation results I

In Figure 7.3 we show the average burst delay versus T_h for different number of TCP connections, N_{tcp}. For a given value of N_{tcp}, it is observed that initially the delay reduces and then increases with increasing values of T_h. This suggests an optimal value of the threshold at which the delay is minimum. At higher values of T_h an increase in the burst delay is observed because a higher value of T_h implies more time is spent in the FACH. The FACH is a low bandwidth channel which has high priority signaling traffic on it. This results in low average bandwidth being shared amongst the TCP connections. For a TCP connection, the switch to DCH is based on its the current buffer size which in turn depends on its current window size. The current window size is incremented whenever an ACK is received by the sender. When a TCP connection is on a low bandwidth link, the window builds up slowly due to delay in receiving an ACK. This slow buildup of the window size results in slow buildup of the current buffer size. As the value of T_h is increased, a TCP connection has to spend more time on the slow FACH before it builds up its buffer size resulting in a higher delay.

We also observe a decrease in the burst delay for small values of T_h. This dip is observed due to the high speed of the DCH and the delay before ACKs can reach the source. Before the source can release more packets to the DCH queue it must receive an ACK from the TCP_{dest}. However, there is a link delay and transmission delay before the ACK is received by the source. If, in this period, the DCH queue becomes empty, the session is switched back to the FACH. As soon as the source receives the ACK it releases packets to the queue thus prompting a switch back to DCH. This redundant switch consumes $500ms$ and thus adds to the delay of the burst. However, if there are enough packets present in the queue such that the queue does not become empty prior to the arrival of the ACK, there is no switch back and forth. The number of packets which ensures that

Table 7.1: Network parameters

N_{dch}	1
R_{dch}	384 kbps
R_{fach}	33 kbps
R_{sig}	24 kbps
T_l	1
D_{sw}	.25 s

Table 7.2: Traffic source parameters

T_{on}	$0.3s$
T_{off}	$5s$
P_{off}	0.33
FS_{avg}	30 kB
k	1.1
Pkt_{size}	320 B

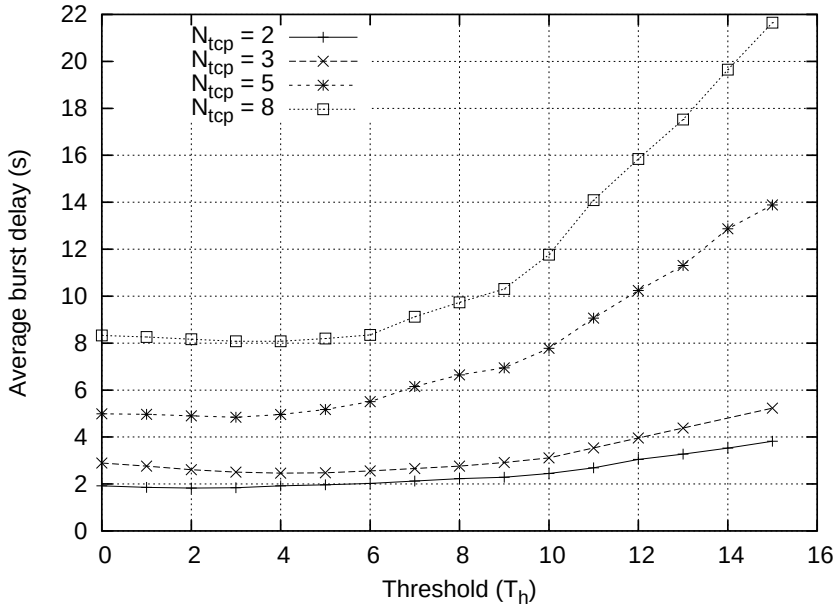


Figure 7.3: Average delay versus T_h. $N_{dch} = 1$, $D_w = 250ms$.

this back and forth switch does not take place is 5 and hence we see an optimal value at $T_h = 4$.

For a given value of T_h, the average burst delay increases as a function of N_{tcp}. As the number of TCP connections on the FACH increase, the available bit-rate for each connection reduces thereby increasing the time taken to increase the window size and hence to exceed the threshold.

In Figure 7.4 we show the throughput as a function of the T_h for different values of the number of TCP connections, N_{tcp}. The throughput first increases and after an optimal value of T_h it decreases. The improvement in throughput at the optimum threshold is more prominent compared to improvement of delay. For a given threshold, the throughput decreases with increasing number of TCP connections.

7.5 Modified threshold policy

In the previous section we observed that the use of the *original* threshold policy resulted in a switch from DCH to FACH even though the source had packets to send. This resulted in a poor delay performance for the policy. In this section we propose a *modified* threshold policy which aims to ameliorate this flaw of the *original* threshold policy. In the *modified* threshold policy, the switch from the FACH to DCH takes place according to the same criterion as the *original* threshold policy. However, when the queue length at SWI_i drops below T_l and the connection i is on DCH we start a timer for a duration T_{out}. If there are packet arrivals at the queue (SWI_i) during this period, we reset the timer to 0. We simulate two policies which decide on the switch back to FACH.

In the *pre-emptive scheme*, if during the timeout period there is another TCP connection which requires a switch to DCH, we initiate a switch back to FACH.

In the *non pre-emptive* scheme, we wait till the end of the timer before we initiate the switch back to FACH. Also, at the end of the timer if there are no other requests for the DCH, we start the timer for a duration of T_{out} thus allowing the connection to remain on DCH.

A timeout value of 0 results in the *original* threshold policy. The use of timeout periods is motivated by the fact that there is a delay before the ACKs reach the source. Hence, even though

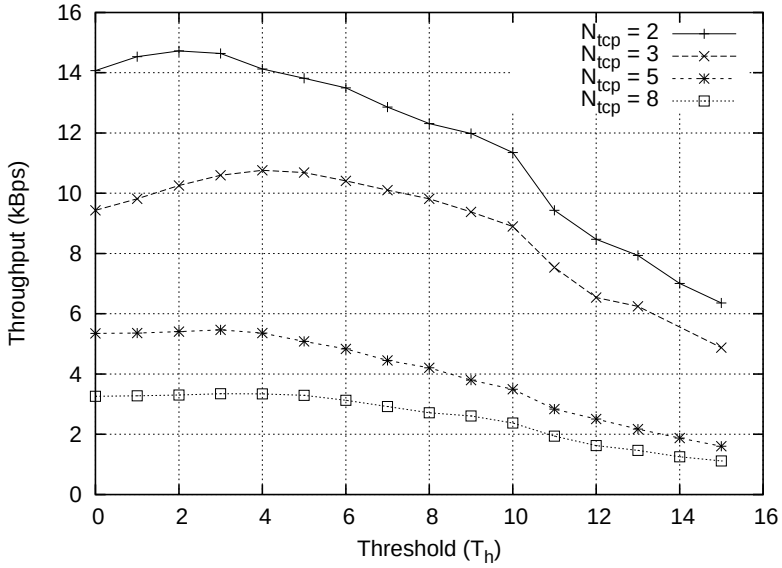


Figure 7.4: Throughput versus T_h. $N_{dch} = 1$, $D_w = 250ms$.

the queue at the base station is empty the source may have packets to send and may be waiting to receive the ACKs before it sends the data packets. The timer, to some extent, waits for the ACKs to reach the source and the source to release the packets.

7.6 Simulation results II

In Figure 7.5 and Figure 7.6 show the average delay performance of the *pre-emptive (pe)* and *non pre-emptive (npe)* schemes of the modified threshold policy. We again note that the case of $T_{on} = 0$ corresponds to the *original* threshold policy. From the figures, we observe that the *modified* threshold policies also have a optimal value of the threshold as was observed in the case of the *original* threshold policy. As compared to the original policy, the modified policy with *pre-emptive* scheme performs marginally better. As the number of TCP connections increases, the probability of a request for a switch also increases. Hence, the *pre-emptive* scheme, in which a switch is initiated as soon as there is a request, performs very close to the original policy when $N_{tcp} = 8$. With $N_{tcp} = 5$, the duration of the idle period on the DCH (i.e., when the timer is on and no requests have been received) is longer and thus the connection is able to receive packets from the source and remain on the DCH and avoid a costly (in terms of time) switch back to FACH. The *non pre-emptive* scheme performs better than both the *original* policy and the *pre-emptive* scheme. The performance gains are close to 20% at $T_h = 0$ and $N_{tcp} = 5$. In the *non-preemptive* scheme, the base station waits till the end of the timer before it initiates a switch. The timeout duration ensures to some extent that the TCP session has ended and thus can be switched back to FACH. Hence, a TCP session which is once switched to DCH remains on the high speed DCH till the end of the session thereby reducing its average delay.

In Figure 7.7 and Figure 7.8 show the performance of the policies when the switching delay, D_w is $500ms$. At a higher switching delay, the optimal threshold shifts to right. As the cost of shifting to DCH and back is high, it is preferable to remain on the FACH when the connections have fewer packets to send. However, we still note that the *non pre-emptive scheme* performs better than the other schemes. With a higher switching delay, the performance gain obtained by using

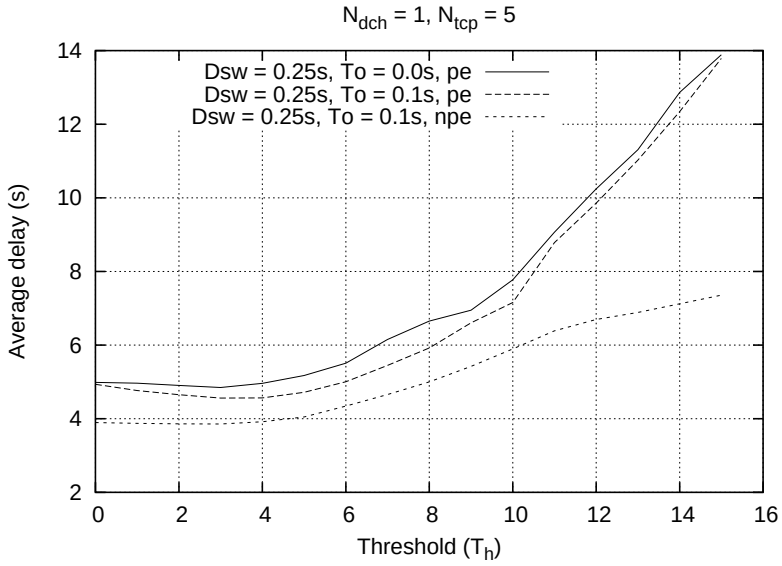


Figure 7.5: Average delay versus T_h. $N_{dch} = 1$,$N_{tcp} = 5$, $D_w = 250ms$.

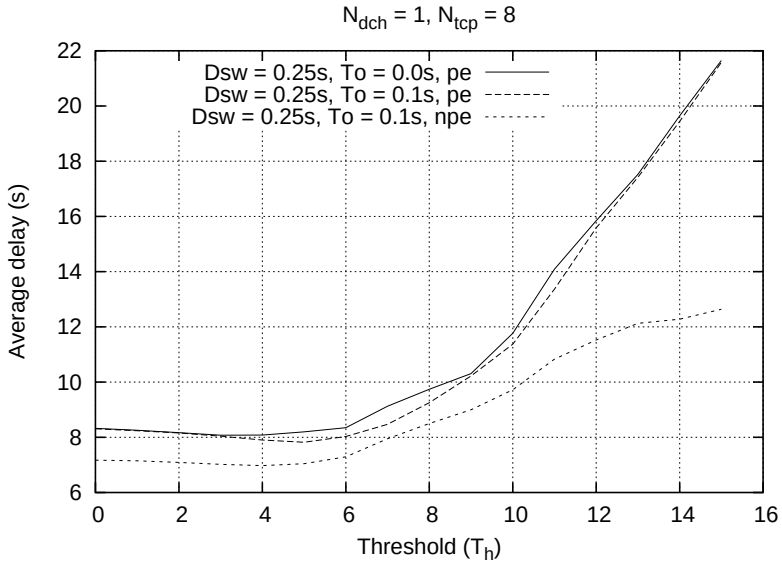


Figure 7.6: Average delay versus T_h. $N_{dch} = 1$,$N_{tcp} = 8$, $D_w = 250ms$.

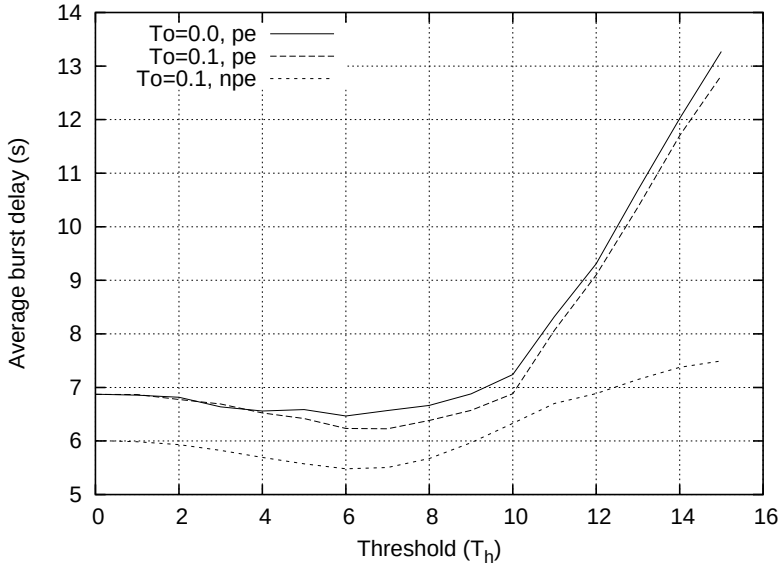


Figure 7.7: Average delay versus T_h. $N_{dch} = 1$,$N_{tcp} = 5$, $D_w = 500ms$.

the optimal threshold is higher and is observed to be around 10% compared to the no threshold case.

In Figure 7.9 we show the average delay with $N_{dch} = 2$, $N_{tcp} = 8$, and $D_w = 250ms$. We observe a similar behavior as in the case of $N_{dch} = 1$.

Also, in general, we observe that there is significant loss in delay performance at threshold values higher than the optimal value.

7.7 Summary and conclusions

We evaluated, through simulations, the performance TCP data transfers on the UMTS downlink. We used a threshold policy to determine which TCP connections should use the high speed dedicated channels. The results indicated the presence of a optimal value of the threshold for minimum burst delay. However, the threshold policy resulted in redundant switches between the FACH and DCH, and therefore in increased average delay for the file transfers. We used a timer on the DCH to prevent these frequent switches. We observed that the use of a timer on the DCH resulted in improved delay performance for the bursts (files). The results indicate that for the simulated parameters the use of a threshold did not significantly improve the delay performance and suggest that use of a timer with a threshold can give improved performance.

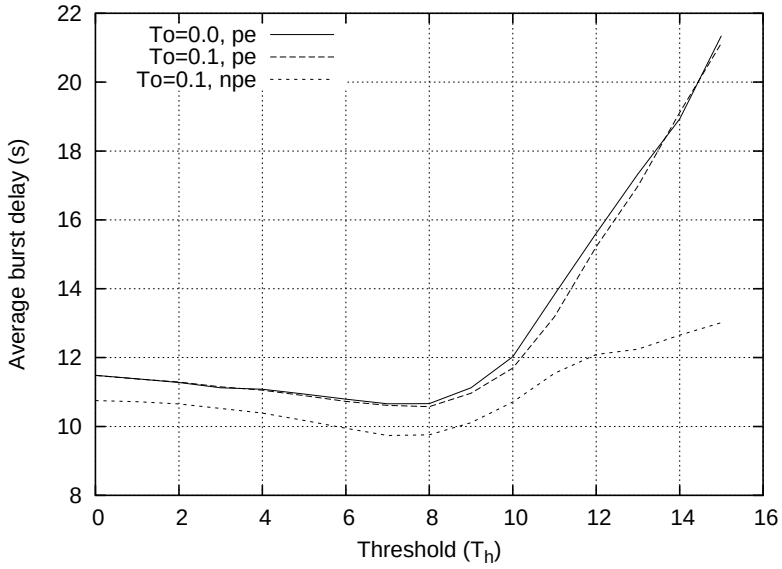


Figure 7.8: Average delay versus T_h. $N_{dch} = 1$,$N_{tcp} = 8$, $D_w = 500ms$.

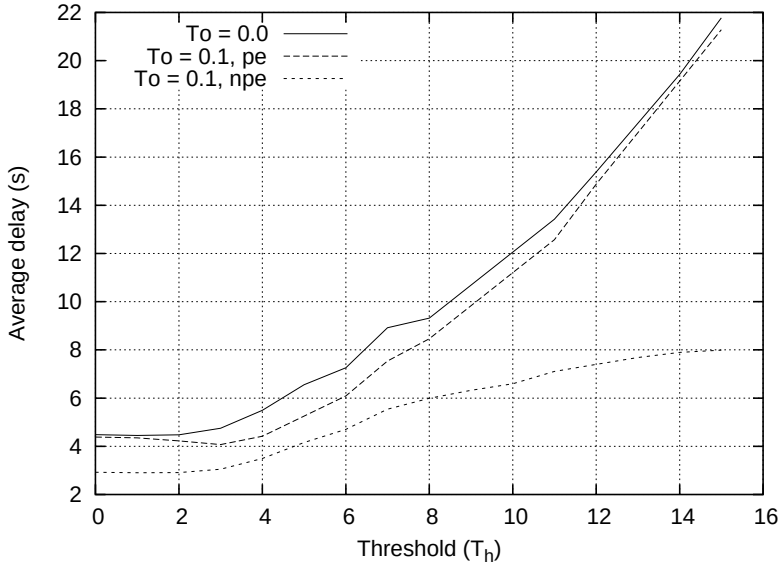


Figure 7.9: Average delay versus T_h. $N_{dch} = 2$,$N_{tcp} = 8$, $D_w = 250ms$.

Chapter 8

Summary, conclusions and perspectives

In Chapter 2, we studied the window-size behaviour of the MIMD algorithm in the presence of three different loss processes. We obtained the expressions for the probability distribution function of the window-size in the presence of window-independent losses for algorithms with both unbounded and bounded window-sizes. For a window-dependent loss process an approximate expression for the throughput was derived when there is an upper bound on the window size. Through this expression, we observed that, in an unbounded system, the throughput was proportional to q^{-1}, where q is the packet loss probability. This proportionality result itself is not new, however we believe that the method employed to arrive at it is original. Although we considered three different loss processes, we did not study the effect of Markovian window-dependent losses. For wireless links, the per packet loss probability is frequently modelled as first-order Markov process. Since wireless technology becomes ubiquitous, the performance analysis of an algorithm in such an environment will be useful. Also, our analysis is restricted to a single instance of the algorithm. In [BH02], the authors study the AIMD algorithm in a multiple session scenario through which they observe that although the sessions obtain the same long term average throughput, the variance in the sending rates is large. Such an analysis of the MIMD algorithm will help to understand the interaction among various sessions.

In Chapter 3, we compared the steady state distribution of the window-size of two different algorithms. We obtained conditions on the loss process so that the algorithms had similar window-size process observed at loss instants. In deriving the steady state behaviour of the window-size, we had made two simplifying assumptions. The first assumption was to model the RTT as a constant. In practice it is possible that there are large buffers[1] in the routers. As the buffer starts to fill up, the changes in the RTTs become more significant. This change in the RTT can alter the window evolution process, and, hence, will affect the throughput of the algorithm. For example, in [KA04] it has been observed that as the buffer fills up, the window-size of the MIMD algorithm grows linearly in time. It will be interesting to study the effect of an RTT that depends on the current window-size. We note that there is a feedback effect wherein an increase in window-size affects the RTT which then affects the increase algorithm. The second simplifying assumption was to model the feedback of the congestion signals as instantaneous feedback. In practice the feedback

[1] By large we mean that the buffer size is such that the time to empty the buffer is of the order of the propagation delay of the outgoing link at router.

arrives after a delay of RTT. As has been mentioned in [Kel03a] and [Sri04], the delay-stability of a congestion control algorithm is affected by its increase algorithm. Since the feedback is delayed, the congestion control algorithm continues with its increase algorithm for an RTT even though the router buffer may be full. Hence, to have a better idea of the effect of the increase algorithm on the overflow of the router buffer, it may be necessary to take into account the feedback delay. As in the perspectives for Chapter 2, another direction to pursue would be to analyse the effect of multiple sessions sharing a link.

The main observation in Chapter 4 was that the fairness index in systems with homogeneous MIMD sessions can be improved by introducing rate-dependent losses at an intensity greater than zero. In systems with heterogeneous sessions, the fairness index can be improved by introducing rate-dependent losses at an intensity greater than a positive number. We modelled the rate ratio process for two sessions only. Firstly, although we expect the condition for the heterogeneous users to hold even for N sessions, the computation of the exact threshold loss intensity will be of help in designing mechanisms which can improve the fairness index. Secondly, we only modelled the rate ratio process and not the sum of the rates. We noted that the fairness index improved by increasing the loss intensity. However, this leads to a decrease in the throughput of the sessions. A joint model for the sum of the rates and rate ratio can hopefully lead to a loss intensity which optimizes a given function of the fairness index and the throughput. We also studied the bandwidth sharing between AIMD and MIMD sessions in systems with synchronous losses only. The bandwidth sharing will depend on the loss process. For example, when losses are only due to a packet error probability, q, the throughput of an AIMD sessions is proportional to $q^{-0.5}$ whereas the throughput of a MIMD session is proportional to q^{-1}. Hence, by considering a loss process with certain proportion of synchronised losses, we can obtain a behaviour different from the purely synchronised or purely asynchronised behaviour.

The analysis of queue-size of a RED-enabled buffer was performed in Chapter 5. By using a probabilistic dynamics for the average queue-size variable, we were able to model the instantaneous and the average queue-sizes as a non-homogeneous QBD process. For a small averaging parameter, we used techniques from singular perturbation theory to obtain the joint distribution of the instantaneous and the average queue-sizes. For the non-limiting case, we noted that the joint distribution can be obtained by using algorithmic techniques. The input traffic sources were assumed to be open-loop, i.e. sources that do not react to packet drops/marks. The RED algorithm was mainly designed to improve the utilization of the link when sources are closed-loop. We believe that an analysis of the queue-size behaviour when the sources are closed-loop, like TCP, can help in choosing the parameters of the RED algorithm. Indeed, a complete model with feedback delays and closed-loop sources may be desirable but intractable but as a first approximation the feedback delay may be neglected.

In Chapter 6, we formulated the joint optimization of energy efficiency and queueing delay as a Markov decision problem. We considered the finite horizon discounted cost, infinite horizon discounted cost and the average cost problems. The main observations of the analysis was the optimality of threshold based policies. Based on the cost functions, the optimal threshold was either a function of the queue-size only or a function of the sum of the queue-size and the depleted battery energy. We made a simplifying assumption that the battery recovers charge linearly with the time it is left idle. An implication of this assumption was that we could, in theory, use a battery to send an infinite number of packets. In practice, the battery charge recovery depends not only on the current available charge but also the entire history of the battery charge-discharge cycle. Hence, the effect on the optimal policy of this dependence may need to be studied. In one of the scenarios we made the simplifying assumption that the battery could not recover its charge.

The optimal policy without this assumption needs to be investigated. Also, we assumed that only batteries with fixed initial charge can be reordered. An extension of this work would be to consider a scenario where the battery charge to be reordered is itself a decision variable. Another possible extension to this model would be to include the effect of wireless channel fluctuations. We assumed that there was a linear relationship between the number of packet sent and the battery charge consumed. However, wireless channels are subject to fading phenomenon due to which packets need to be sent with a higher power in some time slots as compared to power required in the other time slots. Therefore, the device may be reluctant to send packets in a slot if the channel gain is very small. The optimization problem will now include the state of the wireless channel as the third state variable. This state variable can, for example, modulate the slope of the linear proportionality between the packets sent and the charge consumed. The controller will now take a decision based on these three variables.

Through simulations, in Chapter 7 we studied the effect of two channel allocation policies on the average delay of TCP sessions using UMTS channels. Specifically, we studied policies for switching sessions between a slow FACH and a fast DCH, which required a non-negligible switchover time. It was observed that a threshold based policy with hysteresis can reduce the average delay seen by TCP sessions. We do not have a theoretical model for analysing these policies. A theoretical analysis can reveal the exact dependence of the optimal threshold on the switchover times, the transmission rates of DCH and FACH, and the traffic pattern. We believe that such an analysis of the optimal threshold will be of use to UMTS operators in designing their systems. We would like to mention that this problem arose during discussions with the researchers from France Telecom, a mobile telephone operator in France.

Appendix A

Proofs from Chapter 2

A.1 Proof of Proposition 2.7.1

We first note that for $m = 1$, (2.60) gives $\pi(1) = \pi(0)\frac{p}{1-p}$,which is also obtained from (2.59). From (2.59), for $m \geq 1$, $\pi(m)$ satisfies the recursion

$$\pi(m+1) = \frac{\pi(m) - p\pi(m-k)}{1-p}. \tag{A.1}$$

We show that the probabilities given by (2.60) satisfy the above recursion. Let $m = n(k+1)+j$, where $j = m \mod (k+1)$. Let n denote the level of m. There are two cases : a) $j = k$, and b) $j < k$. When $j = k$, $\pi(m)$ and $\pi(m-k)$ are on the same level n. When $j < k$, $\pi(m)$ is on level n and $\pi(m-k)$ is on level $n-1$.

Case a): From (2.60),

$$\begin{aligned}
\pi(n(k+1)+k) &= \frac{\pi(k)}{(1-p)^{n(k+1)}} \\
&\quad \cdot \sum_{i=0}^{n} (-1)^i \left(p^{i-1} S_{i-1,(n-i)(k+1)+k} + p^i S_{i,(n-i)(k+1)+k-1} \right) (1-p)^{ik}, \\
\pi(n(k+1)) &= \frac{\pi(k)}{(1-p)^{n(k+1)-k}} \\
&\quad \cdot \sum_{i=0}^{n} (-1)^i \left(p^{i-1} S_{i-1,(n-i)(k+1)} + p^i S_{i,(n-i)(k+1)-1} \right) (1-p)^{ik},
\end{aligned}$$

Substituting in the RHS of (A.1), we get

$$\begin{aligned}
\pi(n(k+1)+k) - p\pi(n(k+1)) = \frac{\pi(k)}{(1-p)^{n(k+1)}} \cdot \\
&\left[\sum_{i=0}^{n} (-1)^i \left(p^{i-1} S_{i-1,(n-i)(k+1)+k} + p^i S_{i,(n-i)(k+1)+k-1} \right) (1-p)^{ik} \right. \\
&\left. + \sum_{i=0}^{n} (-1)^{i+1} \left(p^i S_{i-1,(n-i)(k+1)} + p^{i+1} S_{i,(n-i)(k+1)-1} \right) (1-p)^{(i+1)k} \right], \\
= \quad & \frac{\pi(k)}{(1-p)^{n(k+1)}} \cdot \\
&\left[\left(p^{i-1} S_{i-1,(n-i)(k+1)+k} + p^i S_{i,(n-i)(k+1)+k-1} \right) (1-p)^{ik} |_{i=0} \right. \\
&+ \sum_{i=1}^{n} (-1)^i \left(p^{i-1} S_{i-1,(n-i)(k+1)+k} + p^i S_{i,(n-i)(k+1)+k-1} \right) (1-p)^{ik} \\
&+ \sum_{i=1}^{n} (-1)^i \left(p^{i-1} S_{i-2,(n-i+1)(k+1)} + p^i S_{i-1,(n-i+1)(k+1)-1} \right) (1-p)^{ik} \\
&\left. + (-1)^{n+1} \left(p^n S_{n-1,0} + p^{n+1} S_{n,-1} \right) (1-p)^{(n+1)k} \right], \\
= \quad & \frac{\pi(k)}{(1-p)^{n(k+1)}} \cdot \left[\left(p^{i-1} S_{i-1,(n-i)(k+1)+k} + p^i S_{i,(n-i)(k+1)+k-1} \right) (1-p)^{ik} |_{i=0} \right. \\
&+ \sum_{i=1}^{n} (-1)^i \left[p^{i-1} (S_{i-1,(n-i)(k+1)+k} + S_{i-2,(n-i)(k+1)+k+1}) \right. \\
&\left. + p^i (S_{i,(n-i)(k+1)+k-1} + S_{i-1,(n-i)(k+1)+k}) \right] (1-p)^{ik} \\
&\left. + (-1)^{n+1} \left(p^n S_{n-1,0} + p^{n+1} S_{n,-1} \right) (1-p)^{(n+1)k} \right],
\end{aligned}$$

The coefficients $S_{i,j}$ satisfy the recursion

$$S_{i,j} = S_{i,j-1} + S_{i-1,j}. \tag{A.2}$$

Also, from (2.63), $S_{-1,j} = \delta_{0,j}, \forall j$, and $S_{0,j} = 1, \forall j \geq 0$. Therefore, for $i = 0$, we can substitute $S_{i-1,(n-i)(k+1)+k}$ by $S_{i-1,(n-i)(k+1)+k+1}$, and $S_{i,(n-i)(k+1)+k-1}$ by $S_{i,(n-i)(k+1)+k}$. Similarly, $S_{i,0} = 1, \forall i$, and $S_{i,-1} = 0, \forall i$. Therefore, we can substitute $S_{n,-1}$ by $S_{n+1,-1}$, and $S_{n-1,0}$ by $S_{n,0}$. The

coefficients of p^{i-1} and p^i can be substituted by the recursion given in (A.2).

$$
\begin{aligned}
= \quad & \frac{\pi(k)}{(1-p)^{n(k+1)}} \cdot \\
& \Bigg[\left(p^{i-1} S_{i-1,(n-i)(k+1)+k+1} + p^i S_{i,(n-i)(k+1)+k}\right)(1-p)^{ik}|_{i=0} \\
& + \sum_{i=1}^{n} (-1)^i \left[p^{i-1} S_{i-1,(n-i)(k+1)+k+1} + p^i (S_{i,(n-i)(k+1)+k}\right](1-p)^{ik} \\
& + (-1)^{n+1} \left(p^n S_{n,0} + p^{n+1} S_{n+1,-1}\right)(1-p)^{(n+1)k} \Bigg], \\
= \quad & \frac{\pi(k)}{(1-p)^{n(k+1)}} \cdot \sum_{i=0}^{n+1} (-1)^i \left[p^{i-1} S_{i-1,(n+1-i)(k+1)} + p^i S_{i,(n+1-i)(k+1)-1}\right](1-p)^{ik} \\
= \quad & (1-p)\pi((n+1)(k+1)) = (1-p)\pi(m+1).
\end{aligned}
$$

This proves case (a).

Case (b) : Since $j < k$, we can write $m = n(k+1) + j - k$ as $(n-1)(k+1) + j + 1$. From (2.60),

$$
\begin{aligned}
\pi(m) = \pi(n(k+1)+j) &= \frac{\pi(k)}{(1-p)^{n(k+1)+j-k}} \\
&\cdot \sum_{i=0}^{n} (-1)^i \left(p^{i-1} S_{i-1,(n-i)(k+1)+j} + p^i S_{i,(n-i)(k+1)+j-1}\right)(1-p)^{ik}, \\
\pi(m-k) = \pi((n-1)(k+1)+j+1) &= \frac{\pi(k)}{(1-p)^{n(k+1)+j-k-k}} \cdot \\
&\sum_{i=0}^{n-1} (-1)^i \left(p^{i-1} S_{i-1,(n-1-i)(k+1)+j+1} + p^i S_{i,(n-1-i)(k+1)+j}\right)(1-p)^{ik},
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\pi(m) - p\pi(m-k) = & \frac{\pi(k)}{(1-p)^{n(k+1)+j-k}} \cdot \\
& \Bigg[\sum_{i=0}^{n} (-1)^i \left(p^{i-1} S_{i-1,(n-i)(k+1)+j} + p^i S_{i,(n-i)(k+1)+j-1}\right)(1-p)^{ik} \\
& + \sum_{i=0}^{n-1} (-1)^{i+1} \left(p^i S_{i-1,(n-1-i)(k+1)+j+1} + p^{i+1} S_{i,(n-1-i)(k+1)+j}\right)(1-p)^{(i+1)k} \Bigg],
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{\pi(k)}{(1-p)^{n(k+1)+j-k}} \cdot \Bigg[\left(p^{i-1} S_{i-1,(n-i)(k+1)+j} + p^i S_{i,(n-i)(k+1)+j-1}\right)(1-p)^{ik}|_{i=0} \\
&\quad + \sum_{i=1}^{n} (-1)^i \left(p^{i-1} S_{i-1,(n-i)(k+1)+j} + p^i S_{i,(n-i)(k+1)+j-1}\right)(1-p)^{ik} \\
&\quad + \sum_{i=1}^{n} (-1)^i \left(p^{i-1} S_{i-2,(n-i)(k+1)+j+1} + p^i S_{i-1,(n-i)(k+1)+j}\right)(1-p)^{ik} \Bigg].
\end{aligned}
$$

Using substitutions similar to that in case (a) we get

$$
\begin{aligned}
&= \frac{\pi(k)}{(1-p)^{n(k+1)+j-k}} \cdot \Bigg[\left(p^{i-1} S_{i-1,(n-i)(k+1)+j+1} + p^i S_{i,(n-i)(k+1)+j}\right)(1-p)^{ik}|_{i=0} \\
&\quad + \sum_{i=1}^{n} (-1)^i \left[p^{i-1} S_{i-1,(n-i)(k+1)+j+1} + p^i (S_{i,(n-i)(k+1)+j}\right](1-p)^{ik} \Bigg] \\
&= \frac{\pi(k)}{(1-p)^{n(k+1)+j-k}} \cdot \sum_{i=0}^{n} (-1)^i \left[p^{i-1} S_{i-1,(n-i)(k+1)+j+1} + p^i S_{i,(n-i)(k+1)+j}\right](1-p)^{ik} \\
&= (1-p)\pi(n(k+1)+j+1) = (1-p)\pi(m+1).
\end{aligned}
$$

This completes the proof.

A.2 Proof of Proposition 2.8.1

Let $\theta = \frac{1-\alpha^{-(k+1)}}{1-\alpha^{-1}}$. Note that θ is less than 1, since α is greater than 1.

The discriminant is given by

$$
\begin{aligned}
c_1^2 - 4c_0c_2 &= (1+(k+1)qW_{max})^2 - 4W_{max}q\theta \\
&= 1 + ((k+1)qW_{max})^2 + (2(k+1) - 4\theta)qW_{max} \\
&> 0.
\end{aligned}
$$

The last inequality is obtained using the fact $k \geq 1$ and $\theta < 1$. Since the discriminant is positive, both the roots, $\mathrm{E}[W]_{1,2}$, are real. Using the fact $c_1 < 0$ and $c_0, c_2 > 0$ along with Descartes' sign rule, we obtain that both the roots are real and positive.

We first show that $\frac{-c_1}{2c_2} > W_{max}$.

$$
\begin{aligned}
\frac{-c_1}{2c_2} &= \frac{1+(k+1)qW_{max}}{2q\theta} \\
&= \frac{1}{2q\theta} + \frac{(k+1)qW_{max}}{2q\theta}.
\end{aligned}
$$

The first term in the above equality is positive. In the second term, we use the fact $\theta < 1$ together with $k \geq 1$, to note that the coefficient of W_{max} is greater than 1. Therefore, we obtain $\frac{-c_1}{2c_2} > W_{max}$. Hence, $\mathrm{E}[W]_1$ in (2.75) does not satisfy the constraint $\mathrm{E}[W] \leq W_{max}$.

Next, we show that $\mathrm{E}[W]_2$ satisfies the given constraint. We denote the dependence of $\mathrm{E}[W]_2$ on q by $f(q)$.

$$\begin{aligned}
\lim_{q\downarrow 0} f(q) &= \lim_{q\downarrow 0}\left(\frac{1}{2q\theta}+\frac{(k+1)W_{max}}{2\theta}\right)-\sqrt{\left(\frac{1}{2q\theta}+\frac{(k+1)W_{max}}{2\theta}\right)^2-\frac{W_{max}}{(q\theta)^2}} \\
&= \lim_{q\downarrow 0}\left(\frac{1}{2q\theta}+\frac{(k+1)W_{max}}{2\theta}\right) \\
&\quad -\sqrt{\left(\frac{1}{2q\theta}+\frac{(k+1)W_{max}}{2\theta}-W_{max}\right)^2+\left(\frac{(k+1)}{\theta}-1\right)W_{max}^2} \\
&= \lim_{q\downarrow 0}\left(\frac{1}{2q\theta}+\frac{(k+1)W_{max}}{2\theta}\right)-\left(\frac{1}{2q\theta}+\frac{(k+1)W_{max}}{2\theta}-W_{max}\right) \\
&= W_{max}. \qquad \text{(A.3)}
\end{aligned}$$

Let $g(q)$ be defined as

$$g(q)=\frac{1}{2q\theta}+\frac{(k+1)W_{max}}{2\theta}.$$

The function $g(q)$ is a continuous and decreasing function of $q\in(0,1]$. Let $h(q)$ be defined as

$$h(q)=\sqrt{(g(q)-W_{max})^2+d_0},$$

where $d_0=\left(\frac{(k+1)}{\theta}-1\right)W_{max}^2>0$. Since $d_0>0$, $h(q)$ is a positive and continuous function of $q\in(0,1]$.

We can rewrite $f(q)$ as

$$f(q)=g(q)-h(q). \qquad \text{(A.4)}$$

Now consider

$$\begin{aligned}
h'(q) &= \frac{(g(q)-W_{max})}{\sqrt{(g(q)-W_{max})^2+d_0}}g'(q), \qquad q\in(0,1] \\
|h'(q)| &= \left|\frac{(g(q)-W_{max})}{\sqrt{(g(q)-W_{max})^2+d_0}}\right||g'(q)|, \qquad q\in(0,1].
\end{aligned}$$

Since $d_0>0$, we obtain

$$|h'(q)|<|g'(q)|. \qquad \text{(A.5)}$$

Since $g(q)$ is a decreasing function of q, $g'(q)<0$. We can rewrite (A.4) as

$$f'(q)=-(|g'(q)|+h'(q)). \qquad \text{(A.6)}$$

Using inequality (A.5) in (A.6), we obtain

$$f'(q)<0.$$

Therefore $f(q)$ is a decreasing function of $q\in(0,1]$. Also, from (A.3), we have $\lim_{q\downarrow 0} f(q)=W_{max}$. Therefore, $f(q)<W_{max}, q\in(0,1]$. Hence, $\mathrm{E}[W]_2$ satisfies the constraint.

A.3 Proof of Proposition 2.8.2

Let $\theta = \frac{1-\alpha^{-(k+1}}{1-\alpha^{-1}}$. We denote the dependence of $\mathrm{E}[W]_2$ on q by $f(W_{max})$.

$$\begin{aligned}
\lim_{W_{max}\to\infty} f(W_{max}) &= \frac{1}{2q\theta}\left[\lim_{W_{max}\to\infty}(1+(k+1)qW_{max}) - \sqrt{(1+(k+1)qW_{max})^2 - 4W_{max}}\right] \\
&= \frac{1}{2q\theta}\left[\lim_{W_{max}\to\infty}(1+(k+1)qW_{max})\right. \\
&\qquad \left. - \sqrt{\left(1-\frac{2\theta}{k+1}+(k+1)qW_{max}\right)^2 - \left(1-\frac{2\theta}{k+1}\right)^2 + 1}\right]. \\
&= \frac{1}{q(k+1)}.
\end{aligned}$$

A.4 Lower and upper bounds on z_0

In this section, we provide a lower bound and an upper bound on the largest root, z_0, of the equation

$$f(z) \stackrel{\text{def}}{=} (1-p)z^{k+1} - z^k + p = 0, \tag{A.7}$$

where $k > 0$ and $0 < p < 1$. It was shown in the proof of Theorem 2.7.2 that the necessary and sufficient condition for $z_0 > 1$ is $p > \frac{1}{k+1}$. We shall assume that this condition is satisfied. The function $f(z)$ may have complex roots but the root of interest, z_0, is real. Hence we shall consider the f as if $f : \mathbb{R} \mapsto \mathbb{R}$.

Proposition A.4.1 *A lower bound for z_0 is given by*

$$z_{lb} = \frac{k}{(k+1)(1-p)}. \tag{A.8}$$

Proof. From the condition $p > \frac{1}{k+1}$, we obtain $z_{lb} > 1$. We note that $f(1) = f(z_0) = 0$ and $z_0 > 1$. Hence, $f(z)$ attains a minima in the interval $(1, z_0)$. We differentiate Equation (A.7) and equate it to zero to obtain

$$\begin{aligned}
f'(z) = (1-p)(k+1)z^k - kz^{k-1} &= 0. \\
\text{Or,} \quad z = \frac{k}{(k+1)(1-p)}.&
\end{aligned}$$

We set $z_{lb} = \frac{k}{(k+1)(1-p)}$. For $z > z_{lb}$,

$$\begin{aligned}
f'(z) &= (1-p)(k+1)z^k - kz^{k-1} \\
&= z^{k-1}((1-p)(k+1)z - k) \\
&> 0.
\end{aligned}$$

The function, $f(z)$, is, therefore, increasing for $z > z_{lb}$. Also, $\lim_{z\to\infty} f(z) \to \infty$. ■

Next, we give an upper bound for z_0.

Proposition A.4.2 *An upper bound for z_0 is given by*

$$z_{ub} = \left(\frac{k+1}{k}\right)^{1-p} \frac{1}{1-p}. \tag{A.9}$$

Proof. First, we show that $f(z_{ub}) > 0$.

$$\begin{aligned} f(z_{ub}) &= z_{ub}^k((1-p)z_{ub} - 1) + p \\ &= z_{ub}^k \left((1-p)\left(\frac{k+1}{k}\right)^{1-p} \frac{1}{1-p} - 1\right) + p \\ &= z_{ub}^k \left(\left(\frac{k+1}{k}\right)^{1-p} - 1\right) + p \\ &> 0. \end{aligned}$$

At $z = 1$,

$$f'(1) = (k+1)(1-p) - k < 0. \tag{A.10}$$

Since $f'(z)$ is negative at $z = 1$ and $f(1) = 0$ and $f(z_0) = 0$, $f(z)$ is negative for $z \in (1, z_0)$ and is positive in (z_0, ∞). Hence, $z_{ub} > z_0$. ■

In Table A.4, we numerically compare the bounds with the true value of z_0.

k	p	z_{lb}	z_0	z_{ub}
2	0.3334	1.0015	1.003	1.967
	0.4	1.111	1.2164	2.1266
	0.7	2.224	3.0927	3.7681
	0.9	6.6889	9.942	10.447
	0.99	68.965	103.438	103.85
3	0.251	1.0013	1.0026	1.6561
	0.3	1.071	1.137	1.747
	0.5	1.50	1.839	2.309
	0.75	3.75	4.967	5.296
	0.9	7.50	9.9909	10.2918
	0.99	74.99	99.99	100.288
5	0.1676	1.0012	1.0024	1.398
	0.2	1.042	1.08	1.447
	0.5	1.668	1.968	2.193
	0.75	3.342	4.007	4.197
	0.9	8.389	10.067	10.251
	0.99	89.285	107.14	107.32

k	p	z_{lb}	z_0	z_{ub}
15	0.0635	1.001	1.0021	1.1343
	0.1	1.0422	1.072	1.178
	0.5	1.8768	2.0019	2.0676
	0.75	3.757	4.008	4.073
	0.9	9.422	10.05	10.1149
	0.99	98.684	105.26	105.32
25	0.0394	1.00104	1.00206	1.081
	0.1	1.0689	1.101	1.1516
	0.5	1.9248	2.0018	2.0414
	0.75	3.853	4.0074	4.0468
	0.99	100.806	104.838	104.877
	0.9986	625	650	650.04
50	0.0206	1.00102	1.00201	1.041
	0.1	1.09	1.111	1.1318
	0.5	1.963	2.002	2.022
	0.9	9.864	10.061	10.081
	0.99	104.384	106.471	106.491
	0.9986	704.22	718.31	718.33

Table A.1: Numerical comparison of the lower bound, the true value, and the upper bound.

Remark A.4.1 *We are in fact interested in* $\log[z_0]/\log[\alpha]$. *Although the bound may be a good approximation for z_0, it may not be a good approximation for* $\log[z_0]$. *For example, for $k = 15$ and $p = 0.0635$ there is a relative error of* 0.1%, *i.e., $(z_0 - z_{lb})/z_0 = 0.001$ in lower bound. However, on a logarithmic scale, there is a relative error of* 50% *for the same value of k and p.*

Appendix B

B.1 Derivation of the Kolmogorov equation in continuous time

Consider a time t, and let $\Pi(t,x) = \text{Prob}(X_t \leq x)$. For some $x \in [X_{min}, X_{max})$, the forward Kolmogorov equation for $\Pi(t,x)$ can be written as

$$\Pi(t+\delta,x) = \Pi(t,y) + \Lambda(X_{max})\delta\phi(t,X_{max})\mathbf{I}(\mathcal{M}) + \int_{u=y}^{g^{-1}(x)} \Lambda(u)\delta d\Pi(t,u) + o(\delta), \tag{B.1}$$

where $y < x$ is such that $x = y + f(y)\delta + o(\delta)$ and $\phi(t,x) = \text{Prob}(X_t = x)$. The first term on the right hand side corresponds to the probability that at time t the process $X_t \leq y < x$ in which case case $X_{t+\delta} \leq x$. The second term corresponds to a loss event which takes the process to a state $X_{t+\delta} \leq x$, and the $o(\delta)$ term contains the probabilities corresponding to two or more jumps in the interval $[t, t+\delta]$. Using Taylor series expansion of f at x, the term $y = x - f(y)\delta + o(\delta)$ can be rewritten as $y = x - f(x)\delta + o(\delta)$. The forward equation is now given by

$$\begin{aligned}\Pi(t+\delta,x) =& \Pi(t, x - f(x)\delta + o(\delta)) + \Lambda(X_{max})\delta\phi(t,X_{max})\mathbf{I}(\mathcal{M}) \\ &+ \int_{u=x-f(x)\delta+o(\delta)}^{g^{-1}(x)} \Lambda(u)\delta d\Pi(t,u) + o(\delta).\end{aligned} \tag{B.2}$$

We now expand the first term on the right hand side in a Taylor series around the point $x - f(x) + o(\delta)$ and the left hand side around the point $(t+\delta)$, we get

$$\begin{aligned}\delta\frac{\partial}{\partial t}\Pi(t,x) + o(\delta) =& -\delta f(x)\frac{\partial}{\partial x}\Pi(t,x) + \Lambda(X_{max})\delta\phi(t,X_{max})\mathbf{I}(\mathcal{M}) \\ &+ \int_{u=x-f(x)\delta+o(\delta)}^{g^{-1}(x)} \Lambda(u)\delta d\Pi(t,u) + o(\delta).\end{aligned} \tag{B.3}$$

We now take the limit $\delta \downarrow 0$, and obtain the integro-differential equation

$$\frac{\partial}{\partial t}\Pi(t,x) + f(x)\frac{\partial}{\partial x}\Pi(t,x) = \Lambda(X_{max})\phi(t,X_{max})\mathbf{I}(\mathcal{M}) + \int_{u=x}^{g^{-1}(x)} \Lambda(u)d\Pi(t,u). \tag{B.4}$$

Assuming $\Pi(t,x)$ converges to a limiting distribution, $\Pi(x)$, as $t \to \infty$, we get the Kolmogorov equation for the steady state distribution as

$$f(x)\frac{d\Pi(x)}{dx} = \Lambda(X_{max})\phi(X_{max})\mathbf{I}(\mathcal{M}) + \int_{x}^{g^{-1}(x)} \Lambda(u)d\Pi(u), \quad x \in [X_{min}, X_{max}). \tag{B.5}$$

B.2 Derivation of the Kolmogorov equation in discrete time

The derivation of this equation follows a similar derivation in [BBBK04]. Let X_n denote the value of X_t at the nth loss instant, i.e., $X_n = X_{\tau_n}$, and let $\widetilde{\Pi}(n,x) = \text{Prob}\,(X_n \leq x)$. The evolution of the process $\{X_n, n \geq 0\}$ can be described by the recursive equation

$$X_{n+1} = g(X_n) + A_{n+1}, \tag{B.6}$$

where A_{n+1} is the increase in the window-size between the nth and the $(n+1)$th loss instants. We note that the increase in window-size in the time interval (τ_n, τ_{n+1}) depends only on the values of $X_t, t \in (\tau_n, \tau_{n+1})$. Therefore, we have the Markovian property

$$\text{Prob}(A_{n+1} \leq y | \{X_i, i \leq n\}) = \text{Prob}(A_{n+1} \leq y | X_n). \tag{B.7}$$

From Equation (B.6) and Equation (B.7), we can write the recursive equation for $\widetilde{\Pi}^c(n,x)$ as

$$\begin{aligned}\widetilde{\Pi}(n+1,x) =& \text{Prob}(A_{n+1} \leq x - g(X_{max}))\tilde{\phi}(n, X_{max})\mathbf{I}(\mathcal{M}) \\ &+ \int_{X_{min}}^{g^{-1}(x)} \text{Prob}(A_{n+1} \leq x - g(u) | X_n = u) d\widetilde{\Pi}(n,u),\end{aligned} \tag{B.8}$$

where $\tilde{\phi}(n,x) = \text{Prob}(X_n = x)$. Let $\mathsf{A}(n+1,y) = \text{Prob}(A_{n+1} \leq y | X_n)$. We now find $\mathsf{A}(n+1,\cdot)$. From Equation (3.3), the probability that no loss occurs in a time interval $[t, t+\delta]$ is given by

$$\text{Prob}(\text{no loss in } [t, t+\delta]) = \exp(-\Lambda(X_t)\delta). \tag{B.9}$$

Let X_t be equal to x. From Equation (3.1), a change in the window-size from x to $x+\epsilon$ is reflected as a change in time from t to $t + \frac{\epsilon}{f(x)} + o(\epsilon)$. From Equation (B.9) we obtain

$$\text{Prob}(\text{no loss in window-interval } [x, x+\epsilon]) = \exp\left(-\frac{\Lambda(x)}{f(x)}\epsilon\right). \tag{B.10}$$

Therefore,

$$\mathsf{A}(n+1, x - g(u)) = 1 - \exp\left(-\int_{g(u)}^{x} \frac{\Lambda(y)}{f(y)} dy\right). \tag{B.11}$$

We assume that the process $\{X_n\}$ converges to a stationary process in the limit $n \to \infty$. Let $\widetilde{\Pi}(x) = \lim_{n\to\infty} \widetilde{\Pi}(n,x)$. From Equation (B.8), we obtain the relation

$$\widetilde{\Pi}(x) = \mathsf{A}(x - g(X_{max}))\tilde{\phi}(X_{max})\mathbf{I}(\mathcal{M}) + \int_{X_{min}}^{g^{-1}(x)} \mathsf{A}(x - g(u)) d\widetilde{\Pi}(u). \tag{B.12}$$

Appendix C

Some existing theorems on stability of Markov chains

Theorem C.0.1 ([FMM95]) *For an irreducible Markov chain $\{s_n\}$ to be null recurrent, it suffices that there exist two functions $f(x)$ and $\psi(x)$, $x \in \mathbb{Z}$, and a finite subset $A \in \mathbb{Z}$, such that the following conditions hold:*

1. *$f(x) \geq 0$, $\psi(x) \geq 0$, $\forall x \in \mathbb{Z}$;*
2. *For some positive α, γ, with $1 < \alpha \leq 2$,*

$$f(x) \leq \gamma[\psi(x)]^{\alpha}, \forall x \in \mathbb{Z}.$$

3. *$\lim_{x_i \to \infty} \psi(x_i) = \infty$ and $\sup_{x \notin A} f(x) > \sup_{x \in A} f(x)$.*
4. *a) $E[f(s(n+1)) - f(s(n)) \,|s(n) = x] \geq 0, \quad x \notin A$;*
   b) $E[\psi(s(n+1)) - \psi(s(n)) \,|s(n) = x] \leq 0, \quad x \notin A$;
   c) $\sup_{x \in X} E[|\psi(s(n+1)) - \psi(s(n))|^{\alpha} \,|s_n = x] = C, < \infty$.

Theorem C.0.2 (Foster, [FMM95]) *An irreducible Markov chain s, on a countable state $\mathbb{Z}$, is ergodic if and only if there exists a positive function $f(\alpha)$, $\alpha \in \mathbb{Z}$, a number $\mu > 0$ and a finite set A such that*

$$E[f(s(n+1)) - f(s(n))|s(n) = i] \leq -\mu, \quad i \notin A, \tag{C.1a}$$

$$E[f(s(n+1))|s(n) = i] < \infty, \quad i \in A. \tag{C.1b}$$

For the next theorem, we shall need the following definition.

Definition 2 *A set C is called a "small" set if there exists an $m > 0$ and a measure ϕ_m, $\phi_m(\mathbb{R}) > 0$, such that*

$$P^m(x, B) \geq \phi_m(B), \quad x \in C, B \in \mathbb{B}(\mathbb{R}). \tag{C.2}$$

Theorem C.0.3 ([MT93]) *For some "small" set $W \in \mathbb{B}(\mathbb{R})$, some constant $h < \infty$, $\mu > 0$, and an extended real-valued function $V : \mathbb{R} \to [0, \infty]$, the Markov chain $\{\mathsf{s}(n)\}$ is stable if*

$$\Delta V(x) := \int_{\mathbb{R}} P(x, dy)V(y) - V(x) \leq -\mu + h\mathbf{I}(x \in W), \tag{C.3}$$

where $P(x, \cdot)$ is the one step transition probability kernel of $\{\mathsf{s}(n)\}$.

Appendix D

Présentation des travaux de thèse

D.1 Introduction

L'Internet est un réseau d'ordinateurs qui permet un échange des données entre ses nœuds. L'idée d'un réseau d'ordinateurs fut conçu au début des années 1960 aux Etats-Unis dans le cadre d'un programme de recherche financé par le département de défense americain. Ce qui a commencé comme une simple expérience entre deux ordinateurs est devenu aujourd'hui un réseau de plus de 300 millions d'ordinateurs répandu sur les six continents.

L'échange de données sur l'Internet s'effectue sur le principe de "commutation par paquets". Un nœud, appelé source, qui souhaite transférer un fichier (c'est-à-dire un ensemble de données), le segmente en paquets qui sont numérotés. Grâce à *Internet Protocol* (IP), les nœuds peuvent tracer le chemin qui mène à une destination donnée. Un chemin est un ensemble de nœuds, appelés routeurs, qui assurent la liaision entre différentes parties de l'Internet. Les paquets traverseront alors le chemin l'un après l'autre avant d'arriver à la destination.

Etant donné que les routeurs sont des nœuds à taux de transmission fini, l'éxpedition de paquets sans contrôle de la part de la source peut résulter en pertes de paquets, non seulement de la source elle-même mais aussi des autres sources qui partagent le même routeur. Afin d'assurer le bon fonctionment du réseau, à la fin des années 1980 *Transmission Control Protocol* (TCP) était proposé [Jac88] et adopté comme le protocole de contrôle de transmission dans la plupart des nœuds de l'Internet. Depuis sa première apparition, le protocole a beaucoup evolué, réflétant en partie la croissance de l'Internet et de son infrastructure.

Contrairement au protocole IP, qui est suivi par tous les nœuds du parcours, TCP n'est suivi que par la source et la destination, assurant ainsi un fonctionnement indépendent du caractère du réseau sur le chemin. Le protocole peut être résumé ainsi: La source envoie une rafale de paquets à la destination. L'ensemble des paquets qui sont émis mais qui ne sont pas encore acquittés, est appelé une fenêtre. Pour chaque paquet reçu, la destination renvoie à la source un acquittement contenant le numero du dernier paquet reçu dans le séquence. Les ressources du réseau sont partagées parmi plusieurs utilisateurs dont le nombre varie en fonction du temps. La source, privée de ces renseignements précis, doit les recueillir à partir des acquittements qu'il recoit. Un acquittement contenant un numero en séquence sert donc à la fois à signaler la sous-utilisation de la capacité du réseau et à signaler la bonne réception d'un paquet de données. La source réagit alors

en envoyant une rafale avec plus de paquets (c'est-à-dire qu'elle augmente son taux de transmission) afin de pouvoir estimer son propre débit équitable. En revanche, un acquittement contenant un numéro de séquence déjà acquitté signale à la source qu'un des paquets n'était pas reçu en séquence et que, en ce moment, il y a très probablement des pertes de paquets, appelés congestion, dans le réseau. La source réagit en renvoyant le paquet perdu et en divisant son taux de transmission par deux. Nous dénotons par "délai de bout en bout" (*Round Trip Time (RTT)*) le temps entre l'émission d'un paquet de données et la réception de l'acquittement correspondant. Dans la phase de *congestion avoidance*, TCP emploie l'algorithme des accroissements additifs et de la décroissance multiplicative (*Additive Increase Multiplicative Increase (AIMD)*). Plus précisément, dans chaque RTT sans pertes de paquets, la source augmente la taille de la fenêtre (c'est-à-dire le taux de transmission) par un paquet (c'est-à-dire d'une façon additive), et lorsqu'une ou plusieurs pertes sont détectées, la source divise la taille de la fenêtre par deux. Cet algorithme d'AIMD a jusqu'ici bien servi à la gestion de congestion dans l'Internet et au partage équitable des ressources de l'Internet entre ses utilisateurs.

Pendant ces dernières années, l'Internet a connu une croissance importante en nombre des utilisateurs et en taux de transmission disponible dans ses liaisons dorsales. Ce dernier a révelé une faiblesse dans l'algorithme AIMD employé par TCP. Nous prenons un exemple pour mieux l'expliquer: supposons qu'une source, qui est en train de transférer des paquets de taille 1000 bytes à un débit de 500 Mbps et un RTT de 200 ms, détecte une perte due à des imperfections dans la fibre de verre. Celle-ci va engendrer une décroissance du débit de la source, et l'algorithme des accroissements additifs mettra plus de cinquante minutes avant de revenir à un débit de 500 Mbps. Or, pendant ces cinquante minutes, le débit serait sous-optimal car la diminution du débit n'était pas due à la congestion et la source aurait pu transmettre à un débit de 500 Mbps sans engendrer des pertes de paquets. Dans l'algorithme AIMD, ce temps de récupération croît linéairement avec le débit auquel la perte était détectée. Avec la croissance du débit qu'une source pourrait obtenir, cette propriété de l'algorithme AIMD ne fut pas considérée comme acceptable. Afin d'améliorer ce comportement sous-optimal de l'algorithme AIMD, plusieurs propositions, notamment FAST [JWL04], WestWood+ [GM04], High-Speed TCP [Flo03], CuBIC [XHR04] et Scalable TCP [Kel03b], ont été faites.

Deux des propriétés désirables d'un algorithme sont l'efficacité et l'équité, et nous les désignons comme des mesures de performance d'un algorithme. Dans cette thèse, nous nous intéresserons aux mesures de performance de Scalable TCP. Le protocole, proposé par T. Kelly, envisage une simple modification de l'algorithme AIMD. Au lieu des accroissements additifs, T. Kelly proposa des accroissements multiplicatifs, des accroissements qui sont plus aggressifs que des accroissements additifs. Alors, Nous avons un algorithme des accroissements multiplicatifs et de la décroissance multiplicative (MIMD). À noter que la modification est simple et donc facilite le déploiement de l'algorithme dans les nœuds de l'Internet. Une des propriétés des accroissements multiplicatifs est que le temps de récupération reste indépendant du débit auquel la perte était détectée. Cette particularité de l'algorithme est désirable surtout lorsque le débit obtenu par une source ne cesse d'augmenter.

Les mesures de performance d'un algorithme peut être évalué en utilisant des outils différents. Par exemple, nous pouvons faire des expériences extensives de cet algorithme dans des conditions très variées. Ainsi, nous pouvons comprendre le comportement de l'algorithme dans des conditions presque réelles. Cependant, l'espace des paramètres d'algorithme est parfois assez grand pour nécessiter un très grand nombre d'expériences avant de retirer des conclusions sur le comportement. Un autre outil, qui reste à notre disposition, est la modélisation et l'analyse stochastiques. Un modèle stochatique de taille de la fenêtre de l'algorithme est exigé par le caractère aléatoire de perte

de paquets. Certes, les modèles peuvent nécessiter des simplifications avant de se prêter à l'analyse. Néanmoins, l'analyse d'un modèle simplifié peut donner des formules qu'on peut corroborer avec des expériences.

D.1.1 Objets et contributions de cette thèse

Dans la première partie de cette thèse nous analyserons les mesures de performance à travers des modèles stochastiques. Plus précisément, nous présenterons trois modèles stochastiques de la taille de la fenêtre d'une session de transfert de données. Chaque modèle est caractérisé par son processus de perte de paquet. D'abord nous étudions un modèle où la probabilité de pertes de paquets dans chaque RTT est indépendante d'un RTT à l'autre et indépendante de la taille de la fenêtre. Ensuite, nous étudions un modèle où chaque paquet de données peut être perdu avec une probabilité constante, et nous obtenons une formule du débit de la source. Une étude similaire de l'algorithme AIMD avait montré que le débit était proportionnel à l'inverse de la racine carrée de la probabilité de perte d'un paquet [OKM96]. Pour l'algorithme MIMD, nous montrons que le débit obtenu par une source est proportionnel à l'inverse de la probabilité de perte d'un paquet. Enfin, nous étudions un modèle où le processus de perte de paquet suit une loi markovienne mais reste indépendant de la taille de la fenêtre. Dans ces trois modèles, nous nous intéresserons au débit obtenu par une source. Les résultats sont obtenus à partir des modèles en temps et en espace discrets.

Nous continuouns cette étude de comportement de la taille de la fenêtre de l'algorithme MIMD mais cette fois-ci en analysant le comportement en temps et en espace continus. Dans cette étude, nous supposons que le processus de perte de paquet est un processus ponctuel qui suit une loi de Poisson dont le paramètre est une fonction de la taille de la fenêtre. Cette étude a pour but d'obtenir des résultats d'équivalence de differents algorithmes. Par exemple, nous montrons que le comportement de l'algorithme AIMD lorsque le paramètre de processus de Poisson est constant est le même que celui de l'algorithme MIMD lorsque le paramètre de processus de Poisson est une fonction linéaire de la taille de la fenêtre. Afin de obtenir ces résultats, nous utiliserons des résultats obtenus en théorie des files d'attente [BBBK04]. Nous montrerons également que le logarithme de la taille de la fenêtre de l'algorithme MIMD est équivalent au temps d'attente dans une file d'attente $G/G/1$. Ceci nous permettra d'obtenir des moments de la taille de la fenêtre.

Ensuite, nous étudions comment deux sources utilisant l'algorithme MIMD partagent un goulot d'étranglement. Les pertes de paquets sont dites synchronisées lorsque les deux sources subissent une perte au même instant. Pour les sources homogènes, c'est-à-dire des sources avec les même paramètres de l'algorithme, Chiu et Jain avaient montré que l'algorithme MIMD était inéquitable lorsque les pertes de paquets étaient synchronisées tandis que l'algorithme AIMD était équitable. Cet argument convainquit Van Jacobson de proposer l'algorithme AIMD pour le protocole TCP [Jac88]. Cependant, les pertes de paquets ne sont pas toujours synchronisées. Nous démontrons que l'indice d'équité augmente lorsque le processus de perte de paquet devient désynchronisé. Pour les sources hétérogènes, dans [XHR04] il était montré que la source avec un plus petit RTT obtenait un débit qui augmentait vers la capacité du goulot d'étranglement lorsque le temps tendait vers l'infini. Nous démontrons que la source avec un plus grand RTT peut aussi obtenir un débit fini si le taux de désynchronisation est strictment supérieur à une quantité qui dépend de la différence entre les deux RTTs et du facteur multiplicatif de l'algorithme MIMD. Les résultats de cette analyse seront corroborrés par des expériences faits par le simulateur *ns-2*. Ensuite, nous étudions le partage de capacité entre plusieurs sources utilisant l'algorithme MIMD et l'algorithme AIMD lorsque le processus de perte de paquet est synchronisé. Nous montrons que le débit obtenu par chaque source AIMD est indépendant de la capacité du goulot d'étranglement

et dépend uniquement des paramètres des deux algorithmes. Le reste de la capacité est partagé entre les sources qui utilisent l'algorithme MIMD.

Dans l'étude précédente nous avions remarqué qu'un certain taux de désynchronisation était nécessaire afin d'améliorer l'indice d'équité lorsque les deux sources étaient hétérogènes. Dans un autre contexte, Floyd et Jacobson proposèrent un algorithme, qui s'appelle *Random Early Detection* (RED), afin d'augmenter le taux de désynchronisation dans les pertes de paquets dans l'Internet. Auparavant, des études avaient démontré qu'à cause de la synchronisation, l'efficacité de l'algorithme AIMD était réduite, et que cette synchronisation avait pour cause le mécanisme de gestion de file d'attente dans les routeurs. Jusque-là, les routeurs utilisaient le mécanisme de gestion *Drop tail* qui acceptait des paquets jusqu'à ce que la file d'attente fût remplie. En revanche, le mécanisme RED rejetait chaque paquet avec une probabilité calculée en fonction d'une moyenne mobile du nombre instantané de paquets dans la file d'attente. Dans cette étude, les mesures de performance d'un algorithme seront la probabilité moyenne de perdre un paquet, la probabilité de perdre un certain nombre de paquets d'une rafale, et le nombre moyen de paquets dans la file d'attente. Nous montrons que, suite à des approximations, nous pouvons obtenir ces mesures numériquement en utilisant des algorithmes de processus de *Quasi-Birth-Death*. Lorsque le paramètre du calcul de la moyenne mobile tend vers zéro, le nombre instantané de paquets dans la file d'attente et la moyenne mobile commencent à évoluer sur deux échelles de temps différentes. Dans ce cas particulier, nous utilisons la méthode de la perturbation singulière afin d'étudier les mesures de performance de l'algorithme RED.

Enfin, nous nous intéresserons aux deux problèmes qui apparaissent dans les études du réseau mobile. Dans le premier problème, nous considérons un système en temps discret. Ce système consiste en un appareil mobile avec une source d'énergie finie et une file d'attente dans laquelle une rafale de paquets arrive dans chaque intervalle de temps. Pour transmettre un paquet, l'appareil doit dépenser un certain nombre d'unités d'énergie. Etant donné le nombre de paquets dans la file d'attente et l'énergie disponible dans la source au début de chaque intervalle de temps, l'appareil doit prendre une des trois décisions suivantes : *(i)* transmettre un certain nombre de paquets, ce qui diminue l'énergie restante, *(ii)* rester inactif, ce qui augment l'énergie restante, et *(iii)* transmettre le maximum possible nombre de paquets et remplacer la source d'énergie, ce qui vaut un coût fini et déterminé. Dans chaque intervalle de temps, un coût proportionnel au nombre de paquets dans la file d'attente et l'énergie restante est généré. L'appareil veut déterminer la politique (c'est-à-dire une suite des décisions) qui minimise les critères suivants: *(i)* coût pénalisé à horizon fini, *(ii)* coût pénalisé à horizon infini, et *(iii)* coût moyen. Nous faisons l'analyse de ce problème dans le cadre des processus de décision markoviens. Dans le cas où la fonction de coût généré dans chaque intervalle possède une dérivée directionnelle constante, nous montrons que le problème peut être divisé en deux sous-problèmes selon le signe de la dérivée directionnelle. Dans les deux sous-problèmes, nous montrons, sans utiliser les caractéristiques de second ordre, *(i)* l'optimalité du contrôle à seuil, *(ii)* la monotonie paramétrique de ce seuil, et *(iii)* l'unicité de ce seuil. Dans le deuxième problème, nous considérons l'évalution de performance de différentes politiques d'ordonnancement des flots TCP sur la voie descendante du réseau *Universal Mobile Telecommunications System (UMTS)*. Ce réseau est la 3ème génération de réseaux mobiles et offre un débit supérieur à 144 kbps à ses utilisateurs. Sur la voie descendante, le réseau dispose de deux canaux différents—le *Dedicated CHannel (DCH)* et le *Forward Access CHannel (FACH)*—pour transmettre les paquets aux utilisateurs. Bien que le DCH offre un débit nettement supérieur par rapport au FACH, le temps de mise en place du canal est aussi nettement supérieur à celui du FACH. À noter que pour un flot de petite taille, le temps de mise en place du canal pourrait être plus important que son temps de transfert, défavorisant ainsi une commutation de flot vers DCH. En revanche, pour un flot de grande taille, le temps

de mise en place pourrait être moins important que son temps de transfert, privilégiant ainsi une commutation vers DCH. En l'absence de renseignement exact sur la taille d'un flot à la station de base, nous proposons deux politiques basées sur le nombre de paquet de chaque flot dans la file d'attente à la station de base. Pour chaque politiques, nous évaluons le délai de transfert d'un flot en fonction de sa taille.

Les modèles présentés dans la plupart des cas sont des modèles markoviens. Soit $\{X_n, n \geq 0\}$, une suite de variables aléatoires. Le processus X_n est une chaîne de Markov si la distribution de X_{n+1} ne dépend que de la valeur de X_n. Ce genre de dépendance fut proposée et étudiée par A.A. Markov au début des années 1900. Nous utiliserons les chaînes de Markov pour étudier la distribution de la taille de la fenêtre de l'algorithme MIMD (chapitres 2 et 3), du processus de rapport des débits de deux sources (chapitre 4), et la distribution conjointe du nombre de paquets dans la file d'attente et de la moyenne mobile (chapitre 5). Nous utiliserons également la théorie des processus de décision markoviens afin de déterminer la politique optimale (chapitre 6).

Dans ce qui suit, nous allons dévélopper les principaux sujets abordés dans les paragraphes précédents.

D.2 Le comportement de la taille de la fenêtre de l'algorithme MIMD en temps discret

Dans cette section, nous analyserons le comportement de la taille de la fenêtre de l'algorithme MIMD en temps discret. Cette analyse nous amènera à la distribution de la taille de la fenêtre en état stationnaire. Notons que la valeur moyenne de la taille de la fenêtre est proportionnelle au débit obtenu par la source. Ainsi, à travers de cette analyse nous essayerons de comprendre le comportement du débit obtenu par une source MIMD en fonction du processus de perte de paquet. Le comportement de la taille de la fenêtre est déterminé en grande partie par le processus de perte de paquets. Dans ce modèle, nous considérons une source qui a un fichier de très grande taille, voire infinie, et qui utilise l'algorithme MIMD pour contrôler la congestion. Dans notre modèle, nous faisons les hypothèses suivantes.

- Pas de *Timeouts*: l'effect d'un timeout est de faire retomber la taille de la fenêtre à sa borne inférieure. Les Timeouts se produisent généralement lorsqu'il y a plusieurs pertes consécutives. Nous supposons que Scalable TCP est utilisé avec SACK, une modification qui peut récupérer plusieurs pertes sans un timeout.
- Au plus une décroissance dans chaque RTT: c'est aussi une des particularités de la modification SACK.
- Une taille infinie du fichier.
- Des RTT constants.

Soit W_n la taille de la fenêtre observée à la fin du $n$ème RTT. La variable W_n évolue selon les règles suivantes

$$W_{n+1} = \begin{cases} \min(\alpha \cdot W_n, W_{max}) & \text{s'il n'y a aucune perte pendant le } n\text{ème RTT,} \\ \max(\beta \cdot W_n, W_{min}) & \text{dans le cas contraire.} \end{cases} \tag{D.1}$$

où $\alpha > 1$ est le paramètre d'accroissement, $\beta < 1$ est le paramètre de décroissance, et W_{min} et W_{max} sont respectivement des bornes inférieure et supérieure de W_n. La borne supérieure est

souvent exigée par la destination à cause d'une taille finie de sa file d'attente tandis que la borne inférieure est nécessaire afin de garantir un certain débit. Notons qu'un taux de transmission fini du goulot d'étranglement impose également une borne supérieure à W_n. La vraie borne supérieure est alors le minimum de ces deux bornes supérieures.

Soit $p(W_n)$ la probabilité de perdre plus qu'un paquet pendant le $n$ème RTT. Nous considérons trois modèles de la fonction $p(W_n)$.

- Modèle (i): La probabilité de perdre plus qu'un paquet dans chaque RTT est une constante p indépendante de W_n. Cette indépendance était observée dans les réseaux étendus [AAB00a], et était étudiée dans [BH02]. Cette indépendance était observée lorsque le nombre de flots tendait vers l'infini et chacun de ces flots traversait plusieurs liaisons [MGT99].

- Modèle (ii): Chaque paquet pourrait être perdu avec une probabilité q. Ce modèle peut servir quand la liaison introduit des pertes à cause des imperfections, c'est le cas, par exemple, des liaisons de fibre de verre. Dans [OKM96], ce modèle était utilisé pour montrer que le débit d'une source qui utilisait l'algorithme AIMD était proportionnel à $1/\sqrt{q}$.

- Modèle (iii): Soit p_n la probabilité de perdre plus qu'un paquet dans le $n$ème RTT. Dans ce modèle la suite p_n est une chaîne de Markov (cf. [AAB00b]).

Un sommaire des analyses et des résultats concernant l'algorithme AIMD est présenté dans [BHCKS04].

D.2.1 Les principaux résultats

- Modèle (i): Sous l'hypothèse du modèle (i), W_n évolue selon les règles suivantes

$$W_{n+1} = \begin{cases} \min(\alpha \cdot W_n, W_{max}) & 1-p, \\ \max(\beta \cdot W_n, W_{min}) & p. \end{cases} \tag{D.2}$$

C'est une récurrence stochastique multiplicative. La suite W_n est une chaîne de Markov en temps discret et en espace borné. Soit $Y_n = \log(W_n)$. Avec cette transformation, la récurrence devient additive, et nous sommes capables d'obtenir les moments de W_n à partir de la transformée de Laplace de Y_n. Quand $k = -\log[\beta]/\log[\alpha]$ est un entier, l'espace aussi devient discret. Soit $\pi(m)$ la probabilité que, en état stationnaire, W_n est égal à m. Nous avons le résultat suivant :

$$\begin{aligned} \pi(m) &= \pi(0)\frac{p}{(1-p)^m} \cdot \sum_{i=0}^{r}(-1)^i\big(p^{i-1}C_{i-1,(r-i)(k+1)+j} + p^i C_{i,(r-i)(k+1)+j-1}\big)(1-p)^{ik}, \\ & \quad m = 0, 1, 2, ..., L, \end{aligned} \tag{D.3}$$

où L est le nombre d'états de W_n, $\pi(0)$ est obtenu par la normalisation $\sum_{m=0}^{L}\pi(m) = 1$, et

$$C_{i,j} = \binom{i+j}{j}. \tag{D.4}$$

– Modèle (ii): Sous l'hypothèse du modèle (ii), W_n évolue selon les règles suivantes :

$$W_{n+1} = \begin{cases} \min(\alpha \cdot W_n, W_{max}) & \text{w.p.} \quad (1-q)^{W_n}; \\ \max(\beta \cdot W_n, W_{min}) & \text{w.p.} \quad 1-(1-q)^{W_n}. \end{cases} \tag{D.5}$$

L'analyse de cette récurrence est difficile. Afin de pouvoir obtenir une formule explicite, nous proposons l'approximation

$$p_n \approx qW_n. \tag{D.6}$$

où p_n est la probabilité de perdre plus qu'un paquet dans le $n$ème RTT. Ceci nous permet d'utiliser l'analyse faite pour le modèle (i). Alors, nous avons

$$\mathrm{E}[W] = \frac{-c_1 - \sqrt{c_1^2 - 4c_2c_0}}{2c_2}. \tag{D.7}$$

où $c_2 = q\frac{1-\alpha^{-(k+1)}}{1-\alpha^{-1}}$, $c_1 = -(1 + (k+1)qW_{max})$, et $c_0 = W_{max}$. Nous notons que, dans cette analyse, $W_{min} = 0$. Lorsque W_{max} tend vers l'infini, nous avons

$$\lim_{W_{max}\to\infty} E[W] = \frac{1}{q(k+1)}. \tag{D.8}$$

Nous remarquons que le débit de l'algorithme MIMD est proportionnel à $1/q$ tandis que celui de l'algorithme AIMD est proportionnel à $1/\sqrt{q}$.

– Modèle (iii): En suivant la méthode d'analyse du modèle (i), nous pouvons obtenir une expression pour la valeur moyenne de la taille de la fenêtre, et cette expression est donnée par Equation (2.89).

Nous avons verifié ces résultats par des expériences faites en utilisant le simulateur *ns-2* [MF].

D.3 Analyse de performance des algorithmes de contrôle de congestion en temps continu

Dans cette section, nous nous intéréssons à la taille de la fenêtre en temps continu. Nous montrons que le processus de la taille de la fenêtre est markovien et écrivons l'équation de Kolmogorov pour la distribution stationnaire de ce processus. Cette équation permet de comparer le comportement des algorithmes et, surtout, d'obtenir des conditions sous lesquelles deux algorithmes auront le même comportement. Une approache similaire a été adopt'ee dans [AABQ01a, KK02, BHCKS04] afin d'analyser l'algorithme AIMD.

Nous pouvons décrire le comportement de la taille de la fenêtre d'un algorithme en spécifiant son algorithme des accroissements, celui de sa décroissance ainsi que le processus de perte de paquets. Soit X_t la taille de la fenêtre à l'instant t. Dans le modèle en temps continu

– Le processus de perte de paquets est ponctuel et suit une loi de Poisson dont le paramètre, $\Lambda(X_t)$, est une fonction de la taille de la fenêtre.

– Dans l'intervalle de temps entre deux pertes de paquets, la taille de la fenêtre croît selon la solution de l'équation différentielle

$$\dot{X}_t = f(X_t), \quad \forall t \in (\tau, \tau + \Delta), \tag{D.9}$$

où $(\tau, \tau + \Delta)$ est un intervalle de temps sans pertes de paquets. Nous notons que la fonction $f(x) = constante$ est une interpolation de l'algorithme des accroissements additifs, et la fonction $f(x) = \alpha x$ est celle de l'algorithme des accroissements multiplicatifs.

– Chaque perte de paquet engendre une décroissance donnée par une fonction $g(X_t)$. Soit τ_n le temps d'arrivé du $n$ème perte de paquet. Ainsi,

$$X_{\tau_n +} = g(X_{\tau_n}). \tag{D.10}$$

À noter que X_t a une borne inférieure, X_{min}, et une borne supérieure, X_{max}.

D.3.1 Les principaux résultats

Un système est défini par les trois fonctions $<f, g, \Lambda>$. Par conséquence de la formulation ci-dessus, le processus $\{X_t\}$ est markovien. Soit $\pi(x)$ la mesure invariante du processus $\{X_t\}$. Dans le cas où $X_{min} = 0$ et $X_{max} = \infty$, La proposition suivante est verifée.

Proposition D.3.1 *Soitent* $<f_1, g_1, \Lambda_1>$ *et* $<f_2, g_2, \Lambda_2>$ *deux systèmes tels que*

1. $g_1(x) = g_2(x) = g(x),\ \forall x,$ *et*
2. $\frac{f_1(x)}{\Lambda_1(x)} = \frac{f_2(x)}{\Lambda_2(x)} = \kappa(x),\ \forall x.$

Alors,

$$\frac{\pi_1(x)}{\pi_2(x)} = C\frac{\Lambda_2(x)}{\Lambda_1(x)} = C\frac{f_2(x)}{f_1(x)}, \quad \forall x, \tag{D.11}$$

où $C = \frac{E[\Lambda_1(X)]}{E[\Lambda_2(X)]}$.

De même façon, dans le cas où $X_{min} = 0$ et $X_{max} < \infty$, nous avons la proposition suivante.

Proposition D.3.2 *Soitent* $<f_1, g_1, \Lambda_1>$ *et* $<f_2, g_2, \Lambda_2>$ *deux systèmes tels que*

1. $g_1(x) = g_2(x) = \beta x,\ \forall x,$ *et*
2. $\frac{f_1(x)}{\Lambda_1(x)} = \frac{f_2(x)}{\Lambda_2(x)}, \forall x.$

Alors,

$$\frac{\pi_1(x)}{\pi_2(x)} = C\frac{f_2(x)}{f_1(x)} = C\frac{\Lambda_2(x)}{\Lambda_1(x)}, \forall x \tag{D.12}$$

$$où \quad C = \frac{\Lambda_1(X_{max})\phi_1(X_{max})}{\Lambda_2(X_{max})\phi_2(X_{max})} \ \ si \ \ \phi_i(X_{max}) > 0, \tag{D.13}$$

$$et \quad \phi(X_{max}) = 1 - \int_0^{X_{max}-} \pi(x)dx. \tag{D.14}$$

Considérons l'exemple suivant : soient $< f_1, g_1, \Lambda_1 >=< \alpha, \beta x, \lambda >$ et $< f_2, g_2, \Lambda_2 >=< \alpha x, \beta x, \lambda x >$. Le premier système représente l'algorithme AIMD et un processus de perte de paquet qui est indépendant de X_t. Le deuxième système représente l'algorithme MIMD et un processus de perte de paquet dont l'intensité dépend linéairement de X_t. Nous constatons que les deux conditions exigées par les deux propositions sont satisfaites. Par conséquence, nous pouvons nous servir de l'analyse d'un système afin d'analyser l'autre.

Par suite, nous analysons l'algorithme MIMD lorsque l'intensité du processus de perte de paquet est une fonction linéaire de X_t. Le système étudié est alors noté $< \alpha x, \beta x, \lambda x >$. Pour ce système, nous pouvons calculer les moments de X_t à l'état stationnaire par la récurrence suivante :

$$E[X] = \frac{\alpha}{\lambda \log[1/\beta]}, \tag{D.15}$$

$$E[X^{j+1}] = \frac{\alpha^j j!}{\lambda^j \prod_{i=1}^{j}(1-\beta^i)} E[X] \quad j = 1, 2, \ldots. \tag{D.16}$$

Dans [BHCKS04], il est noté qu'un processus de perte de paquet à l'intensité de la forme λx est en fait un approximation du cas où la probabilité de perte de paquet est constante. Dans la section précédente, Nous avions obtenu une expression de la valeur moyenne de la taille de la fenêtre en temps discret. En temps continu, nous trouvons que la valeur moyenne est proportionnelle à $1/q$, en accord avec la formule obtenue dans la section précédente.

D.4 Quelques propriétés d'équité de l'algorithme MIMD

Dans les sections précédentes, nous avons étudié le comportement de la taille de la fenêtre d'une source utilisant l'algorithme MIMD. Dans cette section, nous étudions d'abord comment deux sources MIMD se partagent la capacité d'un goulot d'étranglement. Nous étudions ensuite le cas où plusieurs sources utilisant soit l'algorithme MIMD soit l'algorithme AIMD se partagent la capacité d'un goulot d'étrangelement. Nous utiliserons le modèle en temps et en espace continu, décrit en section précédente, pour la taille de la fenêtre d'une source. Des études sur l'équité ont paru entre autres, en [CJ89], [BB01],[HMM00],[Gor04],[XHR04],[KMT98], et [MW00].

Dans un premier temps, nous considérons deux sources homogènes, autrement dit deux sources qui ont les mêmes paramètres de l'algorithme MIMD. Le processus de perte de paquet peut être synchronisé ou désynchronisé. Nous supposons que les pertes se produisent selon un processus de Poisson ponctuel et homogène dont l'intensité est constante. Dans le premier modèle, nous supposons que les pertes sont synchronisées, autrement dit, les deux sources subissent des pertes simultanément. Soit $\theta(t)$ le processus représentant le rapport des débits de deux sources. Dans [CJ89], $\theta(t) = 1$ définit la ligne d'équité. Une des propriétés de l'algorithme des accroissements multiplicatifs est que $\theta(t)$ reste constant entre deux pertes consécutives. Si les pertes sont synchronisées, nous constatons que $\theta(t)$ restera toujours égal à $\theta(0)$. Ainsi, si le partage initial est iniquitable (c'est-à-dire $\theta(t) \neq 1$), il le restera pour toujours. Dans un contexte déterminé, Chiu et Jain avaient proposé un indice d'équité que nous reformulons dans un contexte stochastique de la façon suivante :

$$F_* = \frac{1}{2} + \inf_{\mathsf{x}(0)} \lim_{T\to\infty} \frac{1}{T} \int_0^T \frac{\theta(t)}{1+\theta(t)^2} dt. \tag{D.17}$$

où $\mathsf{x}(0)$ est le vecteur initial des débits. La valeur minimale de cet indice est 0.5 et sa valeur maximale est 1. La valeur maximale de cet indice est atteinte si et seulement si les deux sources

obtiennent le même débit pout tout t. Lorsque le processus de perte de paquet est synchronisé, nous pouvons montrer que l'indice atteint sa valeur minimale. Le processus de perte de paquet est dit dépendant du débit si les pertes sont désychronisées et que la source qui subit la perte est choisie au hasard, avec une probabilité qui croît avec le débit de cette source. De même, le processus de perte de paquet est dit indépendant du débit si la probabilité de choisir une source est indépendante du débit de cette source. Dans [Gor04], l'auteur avait avancé des arguments pour montrer que l'indice d'équité aurait pu être amélioré si le processus de perte de paquet était dépendant du débit.

Dans ce travail, nous présentons un modèle stochastique et obtenons la distribution stationnaire de $\theta(t)$. Soit $\theta(n)$ la chaîne de Markov obtenue par l'observation du processus $\theta(t)$ juste avant une perte. Nous montrons que la chaîne de Markov $\theta(n)$ est récurrente positive si le processus de perte de paquet dépend du débit. Dans ce cas, la valeur de l'indice est supérieure à sa valeur minimale et dépend du facteur multiplicatif de l'algorithme. En revanche, la chaîne $\theta(n)$ est récurrente nulle si le processus de perte de paquet est indépendant du débit, auquel cas l'indice d'équité atteindra sa valeur minimale.

Dans un deuxième temps, nous considérons le cas où les deux sources sont hétérogènes. Le plus souvent, la hétérogénéité est due à une différence entre les RTTs respectifs. Nous montrons que $\theta(n)$ est récurrente positive si le processus de perte de paquet dépend du débit et son intensité est supérieure à une constante qui dépend des paramètres de deux connexions.

Pour finir, nous étudions comment les sources utilisant l'algorithme MIMD et celles utilisant l'algorithme AIMD se partagent la capacité d'un goulot d'étranglement, lorsque le processus de perte de paquet est synchronisé. Nous obtenons des résultats suivants :

- Chaque source qui utilise l'algorithme AIMD obtient un débit indépendant de la capacité du goulot d'étranglement. Ce débit dépend des paramètres des deux algorithmes.

- L'ensemble des sources MIMD obtient la capacité restante.

- Il existe un seuil de la capacité au-dessous duquel une source utilisant l'algorithme AIMD obtiendra un meilleur débit qu'une autre utilisant l'algorithme MIMD.

D.5 L'approche de perturbation singulière pour analyser une file d'attente RED

Nous considérons une file d'attente M/M/1 avec RED. L'algorithme RED peut être décrit ainsi : soit Q_n le nombre de paquets dans la file d'attente juste avant le $n$ième arrivée. L'algorithme consiste à calculer, juste avant chaque arrivée, une moyenne mobile de la façon suivante :

$$q_{n+1} = (1 - \alpha)q_n + Q_{n+1}, \tag{D.18}$$

où $0 < \alpha < 1$. Le $n$ième paquet arrivé est rejeté avec une probabilité qui dépend de q_n.

Plusieurs travaux ont porté sur l'analyse d'une file d'attente RED [MGT00, BMB00, HMTG01, KLVK01, SVL02]. Nous proposons un analyse stochastique lorsque le paramètre α tend vers zéro. Il se trouve que le processus conjoint (q_n, Q_n) est une chaîne de Markov. Afin de faciliter l'analyse, nous proposons l'approximation suivante :

$$\hat{q}_{n+1} = \hat{q}_n + C_n, \tag{D.19}$$

où

$$C_n = \begin{cases} 1 & \text{avec une probabilité } \alpha(mQ_{n+1} - \hat{q}_n)^+ ; \\ -1 & \text{avec une probabilité } \alpha(\hat{q}_n - mQ_{n+1})^+ ; \\ 0 & \text{sinon} , \end{cases}$$

où la notation $x^+ = \max(x, 0)$ a été utilisé.

Grâce à cette approximation, le processus conjoint $(\hat{q}_n, Q_n)$ devient un processus QBD. Pour calculer la distribution stationnaire d'un processus QBD, les algorithmes décrits dans [GJL84] et [LR99] peuvent être utilisés. Dans le cas où le paramètre α tend vers zéro, $\hat{q}_n$ évolue selon une échelle de temps plus lente que celle de Q_n. Cette séparation des échelles de temps nous permet d'utiliser une perturbation singulière pour obtenir la distribution jointe et les autres mesures de performance. À noter que pour obtenir la distribution stationnaire de $\hat{q}_n$, nous pouvons supposer que Q_n se trouve déjà à l'état stationnaire. Pour calculer la valeur moyenne du nombre de paquets dans la file d'attente, nous proposons une formule approximative mais facile à calculer.

D.6 Contrôle optimal du délai et de l'énergie dans un appareil mobile

Soit un appareil mobile qui veut transmettre des paquets en temps discret. Dans chaque intervalle de temps, une rafale d'un nombre aléatoire de paquets arrive dans la file d'attente de l'appareil. L'appareil a une source d'énergie finie à sa disposition. Soit M le nombre maximale d'unités d'énergie que la source d'énergie peut contenir. Tout paquet transmis consomme un nombre fixe d'unités d'énergie. Soient p_n et x_n, respectivement, le nombre restant d'unités d'énergie et le nombre de paquets dans la file d'attente au début du $n$ième intervalle. Etant donné p_n et x_n, au début du $n$ième intervalle, l'appareil doit prendre un des trois décisions suivantes :

- transmettre ne rien, ce qui augmenterait le nombre restant d'unités d'énergie (le nombre d'unités d'énergie est toujours borné par M);
- transmettre un certain nombre de paquets, ce qui diminuerait le nombre de paquets dans la file d'attente mais aussi le nombre restant d'unités d'énergie. Le nombre de paquets pouvant être transmis est borné par le minimum de p_n et x_n;
- tranmettre $\min(p_n, x_n)$ et remplacer la source d'énergie avec une autre pleine, ce qui induit un coût fixe.

Dans chaque intervalle de temps, un coût $h(x_n) + g(p_n)$ est généré. L'appareil veut minimiser (i) le coût penalisé à horizon fini, (ii) le coût penalisé à horizon infini, et (iii) le coût moyen. Le processus du nombre de paquets dans chaque rafale est une suite de variables aléatoires i.i.d. telle que le nombre moyen de paquets dans une rafale est inférieur à M. Sous cette hypothèse, le problème peut être formulé comme un processus de décision markovien. Nous montrons que la politique optimale pour minimiser le coût moyen est celle du contrôle à seuil. À cette fin, nous faisons l'hypothèse que le signe de $h'(x) + g'(p)$ est indépendant de x et de p pout tout (x, p). Sous cette hypothèse, nous montrons que l'appareil soit transmet rien soit il transmet $\min(x_n, p_n)$, et que le choix dépend du signe de la derivée directionelle, $h'(x) + g'(p)$. Nous pouvons alors réduire l'espace des décisions, et, par récurrence, nous montrons que la politique du contrôle à seuil est optimale pour minimiser le coût penalisé à horizon fini et le coût penalisé à horizon infini. Enfin, nous nous servons d'un théorème de Schäl [Sch93] pour montrer l'optimalité du contrôle à seuil pour minimiser le coût moyen.

D.7 Politiques d'ordonnancement des flots TCP dans l'UMTS

Sur la voie descendante, la station de base dispose de deux canaux – FACH et DCH. Nous faisons l'hypothèse que le canal FACH offre un débit maximal de 32 kbps, et le canal DCH offre un débit constant de 384 kbps. Par ailleurs, le temps de mise en place de DCH pour chaque flot est de l'ordre de 250 ms tandis que celui de FACH est négligeable. Le canal DCH est un canal dedié au flot qui l'utilise tandis que le débit du canal FACH pourrait être partagé entre plusieurs sources. D'ailleurs, le trafic de signalisation est transmis sur FACH, et ce trafic est prioritaire par rapport au trafic des données. La station de base doit choisir quel le flot peut transmettre sur DCH. À noter qu'il n'est pas rentable de commuter les flots de petite taille sur DCH. Puisque la station de base ne peut connaître à l'avance la taill d'un flot, nous proposons une politique basée sur le nombre de paquets d'un flot dans la file d'attente à la station de base. À savoir que chaque flot qui arrive est accordé une file d'attente virtuelle à la station de base. Dans cette politique, chaque flot commence à transmettre sur FACH. Si le nombre de paquets d'une source dans sa file d'attente dédiée dépasse un seuil et le DCH est disponible, alors le flot est commuté vers DCH. En faisant des expériences sur *ns-2*, nous avons obtenu le temps de transfert moyen d'un flot en fonction de la seuil. Nous avons constanté qu'il existe un seuil optimal qui minimise le temps de transfert moyen.

D.8 Perspectives

Dans les chapitres 2 et 3, nous avons étudié le comportement de la taille de la fenêtre de l'algorithme MIMD. Nous avons constaté que le débit obtenu est proportionnel à l'inverse de la probabilité de perte d'un paquet. L'analyse fut faite pour trois processus différents de perte de paquet. Nous pourrions étudier le comportement lorsque la probabilité de perte d'un paquet est un processus markovien. Ce processus est souvent utilisé pour des liaisons sans-fil. Notre étude considére le cas d'une seule source. Dans [BH02], une étude est faite dans le cas où plusieurs sources utilisant l'algorithme AIMD. Ce genre d'étude pourrait nous aider à mieux comprendre la dynamique du système lorsqu'il y a une interaction entre les sources. Notre étude fut réalisée sous certaines hypothèses. Une de ces hypothèses est que RTTs sont constants. Or, en général, les RTTs dépendent de la taille des files d'attente dans les routeurs. Par conséquence, ils dépendent de la taille de la fenêtre de chaque source. Nous pourrions étudier le comportement dans pareille situation.

Dans le chapitre 4, nous avons constaté que l'indice d'équité dans le cas de sources homogènes augmente lorsque le processus de perte de paquet dépend du débit, et dans le cas de sources hétérogènes l'intensité de ce processus doit être supérieure à une constante qui dépend des paramètres de l'algorithme. L'indice d'équité croît avec l'intensité. À noter que nous avons considéré le rapport des débits de deux sources et non pas leur débit cumulé. Lorsque le processus de perte de paquet dépend du débit, une intensité forte diminue l'efficacité. Il existe donc un compromis entre l'efficacité et l'indice d'équité. Nous pourrions étudier ce compromis afin de trouver l'intensité optimale. Dans le chapitre 5, nous avons étudié une file d'attente RED lorsque les flux d'arrivée sont en boucle ouverte. La plupart des flux dans l'Internet sont en boucle fermée (les flux qui utilisent TCP, par exemple). Alors, il est possible d'analyser la file d'attente lorsque les flux sont en boucle fermée. Ceci permettra de choisir de bonnes valeurs pour les paramètres de RED. L'analyse presenté dans le chapitre 6 a été conduite sous certaines hypothèses. Une de ces hypothèses est que la croissance de l'énergie restante est un processus sans mémoire. Un modèle plus vraisemblable devrait prendre en compte la dépendance au passé de la source d'énergie. Dans le chapitre 7, nous avons étudié la performance de deux algorithmes d'ordonnancement par simulations. Nous avons constaté qu'il existe un seuil de nombre de paquet dans la file d'attents

au-dessous du quel il était optimal de commuter une source de FACH à DCH. Il serait intéressant de développer un modèle permettant de calculer analytiquement le seuil optimal. Les operateurs mobiles porraient se servir d'une telle analyse afin d'offrir un meilleur service à leurs utilisateurs.

Appendix E

List of Acronyms

ACK	ACKnowledgement
AIMD	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
DCH	Dedicated CHannel
FACH	Forward Access CHannel
MIMD	Multiplicative Increase Multiplicative Decrease
PASTA	Poisson Arrivals See Time Averages
QBD	Quasi-Birth-and-Death
RED	Random Early Detection
RTT	Round Trip Time
SACK	Selective ACKnowledgement
TCP	Transmission Control Protocol
UMTS	Universal Mobile Telecommunications System

Bibliography

[AAB00a] E. Altman, K. Avrachenkov, and C. Barakat. A Stochastic Model of TCP/IP with Stationary Random Losses. In *Proc. of the ACM SIGCOMM*, September 2000.

[AAB00b] E. Altman, K. E. Avrachenkov, and C. Barakat. TCP in Presence of Bursty Losses. In *Proc. of ACM SIGMETRICS*, pages 124–133, 2000.

[AAB+05] E. Altman, K. Avrachenkov, C. Barakat, A. A. Kherani, and B. J. Prabhu. Analysis of MIMD congestion control algorithm for high speed networks. *Computer Networks*, 48:972–989, 2005.

[AABQ01a] E. Altman, K. Avrachenkov, C. Barakat, and R. Núñez Queija. State-dependent M/G/1 Type Queueing Analysis for Congestion Control in Data Networks. In *Proc. of IEEE INFOCOM*, April 2001.

[AABQ01b] E. Altman, K. Avrachenkov, C. Barakat, and R. Núñez Queija. TCP modeling in the presence of nonlinear window growth. In *Proc. of ITC-17*, 2001.

[AAKP05] E. Altman, K. Avrachenkov, A. A. Kherani, and B. J. Prabhu. Performance Analysis and Stochastic Stability of Congestion Control Protocols. In *Proc. of IEEE INFOCOM*, Miami, USA, March 2005.

[AAP04] E. Altman, K. Avrachenkov, and B. J. Prabhu. A Singular Perturbation Approach to Analysing a RED Queue. In *Proc. of International Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs'04)*, Ilkley, UK, July 2004.

[AAP05] E. Altman, K. Avrachenkov, and B. J. Prabhu. Fairness in MIMD Congestion Control Algorithms. In *Proc. of the IEEE INFOCOM*, 2005.

[AFH02] K.E. Avrachenkov, J.A. Filar, and M. Haviv. Singular Perturbations of Markov Chains and Decision Processes. A Survey. In E.A. Feinberg and A. Shwartz, editors, *Handbook of Markov Decision Processes: Methods and Applications*, International Series in Operations Research and Management Science, pages 113–153. Kluwer Academic Publishers, January 2002.

[ALLY01] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active Queue Management. *IEEE Network*, 15(3):48–53, May/June 2001.

[APS99] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581, Standards Track, April 1999.

[Asm03] S. Asmussen. *Applied probability and queues*. Springer, 2003.

[AT03] F. Anjum and L. Tassiulas. Comparative study of various TCP versions over a wireless link with correlated losses. *IEEE/ACM Trans. Netw.*, 11(3):370–383, 2003.

[BB87] F. Baccelli and P. Bremaud. *Palm Probabilities and Stationary Queues*. Springer, 1987.

[BB00] D. Bansal and H. Balakrishnan. TCP-friendly Congestion Control for Real-time Streaming Applications. Technical Report MIT-LCS-TR-806, MIT Lab. for Comp. Science, 2000. `http://citeseer.ist.psu.edu/bansal00tcpfriendly.html`.

[BB01] D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms. In *Proc. of the IEEE INFOCOM*, April 2001.

[BB03] F. Baccelli and P. Bremaud. *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences*. Springer, 2nd edition, 2003.

[BBBK04] R. Bekker, S.C. Borst, O.J. Boxma, and O. Kella. Queues with Workload-Dependent Arrival and Service Rates. *Queueing Sys.*, 46(3-4):537–556, March 2004.

[BCD97] M. Bardi and I. Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman Equations*. Birkhäuser, 1997.

[Ber95] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[BH02] F. Baccelli and D. Hong. AIMD, Fairness and Fractal Scaling of TCP Traffic. In *Proc. of the IEEE INFOCOM*, April 2002.

[BHCKS04] A. Budhiraja, F. Hernández-Campos, V.G. Kulkarni, and F. D. Smith. Stochastic Differential Equation for TCP Window Size: Analysis and Experimental Validation. *Prob. in the Engg. and Informational Sciences*, 18:111–140, 2004.

[BKP05] V. S. Borkar, A. A. Kherani, and B. J. Prabhu. Control of Buffer and Energy of a Wireless Device: Closed and Open Loop Approaches. In *Proc. of WiOpt*, Trento, Italy, April 2005.

[BMB00] T. Bonald, M. May, and J. Bolot. Analytic Evaluation of RED Performance. In *Proc. of the IEEE INFOCOM*, 2000.

[Bor03] S. C. Borst. User-Level Performance of Channel-Aware Scheduling Algorithms in Wireless Data Networks. In *Proc of IEEE INFOCOM*, 2003.

[BP95] L. S. Brakmo and L. L. Peterson. TCP Vegas: end-to-end congestion avoidance on a global internet. *IEEE Jl. on Selected Areas in Comm.*, 13(8):1465–1480, October 1995.

[Bra89] R. Braden. RFC 1122: Requirements for Internet Hosts – Communication Layers, October 1989.

[CB97] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997.

[CCLS03] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun. *Nonlinear Estimation and Classification*, volume 171 of *Lecture Notes in Statistics*, chapter Internet Traffic Tends Toward Poisson and Independent as the Load Increases, pages 83–109. Springer, 2003.

[CF96] F. Camilli and M. Falcone. *Nonsmooth Analysis and Geometric Methods in Deterministic Optimal Control*, chapter Approximation of optimal control problems with state constraints: estimates and applications. Springer, 1196.

[CGB04] S. Cui, A. J. Goldsmith, and A. Bahai. Energy-constrained modulation optimization. *To appear in IEEE Trans. on Wireless Communications,*, 2004.

[CJ89] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[CJOS01] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith. Tuning RED for Web Traffic. *IEEE/ACM Transactions on Networking*, 9(3):249–264, June 2001.

[CR99] C. F. Chiasserini and R. R. Rao. Pulsed battery discharge in communication devices. In *Proc. of ACM/IEEE MobiCom '99*, pages 88–95. ACM Press, 1999.

[CR01a] C.-F. Chiasserini and R. R. Rao. Energy Efficient Battery Management. *IEEE JSAC Wireless Series*, 19(7):1235–1245, July 2001.

[CR01b] C.-F. Chiasserini and R. R. Rao. Improving Battery Performance by Using Traffic Shaping Techniques. *IEEE JSAC Wireless Series*, 19(7):1385–1394, July 2001.

[CSA01] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP Latency. In *Proc. of INFOCOM*, 2001.

[Dav93] M. Davis. *Markov models and optimization*. Chapman Hall, 1993.

[DC99] F. Dufour and O.L.V. Costa. Stability of Piecewise Deterministic Markov Processes. *SIAM Journal on Control and Optimization*, 37(5):1483–1502, 1999.

[FDN94] T. F. Fuller, M. Doyle, and J. S. Newman. Relaxation phenomena in lithium-ion insertion cells. *J. Electrochem. Soc.*, 141:982–990, 1994.

[FGF+99] R. Fielding, J. Gettys, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Standards Track, June 1999.

[FGS01] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, 2001. `http://www.icir.org/floyd/papers.html`.

[FJ93] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[FKSS02] W. Feng, D. Kandlur, D. Saha, and K. Shin. The Blue Queue Management Algorithms. *IEEE/ACM Transactions on Networking*, 10(4):249–264, August 2002.

[Flo94] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24(5):10–23, October 1994.

[Flo97] S. Floyd. RED: Discussions of Setting Parameters, November 1997. Available at `http://www.icir.org/floyd/red.html`.

[Flo00] S. Floyd. Recommendation on using the gentle variant of RED, March 2000. Available at `http://www.icir.org/floyd/red.html`.

[Flo03] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003.

[FMM95] G. Fayolle, V.A. Malyshev, and M.V. Menshikov. *Topics in the Constructive Theory of Countable Markov Chains*. Cambridge University Press, 1995.

[FRS01] S. Floyd, S. Ratnasamy, and S. Shenker. Modifying TCP's Congestion Control for High Speeds. Rough draft, November 2001. `http://www.icir.org/floyd/papers/hstcp.ps.Nov01`.

[GJL84] D.P. Gaver, P.A. Jacobs, and G. Latouche. Finite Birth-and-death Models in Randomly Changing Environments. *Adv. Appl. Probability*, 16:715–731, 1984.

[GK98] C. M. Goldie and C. Kluppelberg. *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, chapter Subexponential Distributions, pages 435–459. Birkhäuser, Boston, 1998.

[GKS03] Munish Goyal, Anurag Kumar, and Vinod Sharma. Power Constrained and Delay Optimal Policies for Scheduling Transmissions over a Fading Channel. In *Proc. of the IEEE INFOCOM*, 2003.

[GM02] L. A. Grieco and S. Mascolo. Westwood TCP and easy RED to improve Fairness in High Speed Networks . In *Proc. of IFIP/IEEE PfHSN*, April 2002.

[GM04] L. A. Grieco and S. Mascolo. Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. *SIGCOMM Comput. Commun. Rev.*, 34(2):25–38, 2004.

[Gor04] S. Gorinsky. Feedback Modeling in Internet Congestion Control. In *Proc. of the NEW2AN*, February 2004. Also see a techincal report at `http://www.arl.wustl.edu/~gorinsky/TR2002-39.ps`.

[GS99] M. E. Gomez and V. Santoja. Analysis of Self-similarity in I/O Workload using Structural Modeling. In *Proc. of* 7^{th} *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, March 1999.

[GY95] P. Glasserman and D.D. Yao. Stochastic Vector Difference Equations with Stationary Coefficients. *J. Appl. Prob.*, 32:851–866, 1995.

[Haj84] B. Hajek. Optimal Control of Two Interacting Service Stations. *IEEE Transactions on Automatic Control*, AC-29:491–499, 1984.

[HMM00] G. Hasegawa, M. Murata, and H. Miyahara. Fairness and Stability of congestion control mechanisms of TCP. *Telecommunication Systems*, pages 167–184, November 2000.

[HMTG01] C. Hollot, V. Misra, D. Towsley, and W. Gong. A Control Theoretic Analysis of RED. In *Proc. of the IEEE INFOCOM*, 2001.

[HT01] H. Holma and A. Toskala. *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications.* J. Wiley and Sons, 2001.

[HW79] G. H. Hardy and E. M. Wright. *An Introduction to the theory of numbers.* Clarendon press, 1979.

[Jac88] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988.

[JMM04] S. Jayashree, B. S. Manoj, and C. Siva Ram Murthy. On using battery state for medium access control in ad hoc wireless networks. In *Proc. of MobiCom '04*, pages 360–373, 2004.

[JWL04] C. Jin, D. X. Wei, and S. H. Low. FAST TCP: Motivation, Architecture, Algorithms, Performance. In *Proc. of the IEEE INFOCOM*, March 2004.

[KA04] R. El Khoury and E. Altman. Analysis of Scalable TCP. In *Proc. of the International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETS)*, July 2004.

[KD92] H. J. Kushner and P. G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time.* Springer, 1992.

[Kel03a] F. Kelly. Fairness and stability of end-to-end congestion control. *European Journal of Control*, 9:159–176, 2003.

[Kel03b] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *Computer Comm. Review*, 33(2):83–91, April 2003.

[KH98] Anurag Kumar and Jack Holtzman. Performance Analysis of Versions of TCP in a Local Network with a Mobile Radio Link. *Sadhana: Indian Academy of Sciences Proceedings in Engg. Sciences*, 23(1):113–129, February 1998.

[KHR02] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *Proc. of the ACM SIGCOMM*, pages 89–102, 2002.

[KK02] A. A. Kherani and A. Kumar. Stochastic Models for Throughput Analysis of Randomly Arriving Elastic Flows in the Internet . In *Proc. of IEEE INFOCOM*, June 2002.

[Kle75] L. Kleinrock. *Queueing Systems Volume I: Theory.* Wiley & Sons, 1975.

[KLVK01] P. Kuusela, P. Lassila, J. Virtamo, and P. Key. Modeling RED with Idealized TCP Sources. In *Proceedings of IFIP ATM & IP 2001*, pages 155–166, Budapest, Hungary, June 2001.

[KMFB04] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A Nonstationary Poisson View of Internet Traffic. In *Proc. of IEEE INFOCOM*, March 2004.

[KMT98] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[KR88] B. W. Kernighan and D. M. Ritchie. *The C programming language*. Prentice Hall, 2nd edition, 1988.

[Kra99] S. G. Krantz. *Handbook of Complex Variables*. Birkhauser, 1999.

[KRV02] E. Kuumola, J.A.C. Resing, and J. Virtamo. Joint Distribution of Instantaneous and Averaged Queue Length in an M/M/1/K System. In P. Tran-Gia and J. Roberts, editors, *Proc. of the 15th ITC Specialist Seminar "Internet Traffic Engineering and Traffic Management"*, pages 58–67, July 2002.

[KSK76] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. Springer-Verlag, New York, 2nd edition, 1976.

[Kum98] Anurag Kumar. Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Trans. Netw.*, 6(4):485–498, 1998.

[LM97] T. V. Lakshman and Ui. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Trans. Netw.*, 5(3):336–350, 1997.

[LR99] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM Series on Statistics and Applied Probability. SIAM, 1999.

[LTWW94] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions Networking*, 2(1):1–15, 1994.

[LV00] P. Lassila and J. Virtamo. Modeling the dynamics of the RED algorithm. In *Proceedings of Quality of Future Internet Services (QofIS)*, pages 28–42, Berlin, Germany, September 2000. Sringer-Verlag.

[LZ96] D. Liu and Y. Q. Zhao. *Matrix-Analytic Methods in Stochastic Models*, chapter Determination of explicit solution for a general class of Markov processes, pages 343–357. Marcel Dekker, 1996.

[Mah97] B. A. Mah. An Empirical Model of HTTP Network Traffic. In *Proc. of IEEE INFOCOM*, April 1997.

[MF] S. McCanne and S. Floyd. ns: Network Simulator. Available at `http://www.isi.edu/nsnam/ns/`.

[MGG+04] M. A. Marsan, M. Garetto, P. Giaccone, E. Leonardi, E. Schiattarella, and A. Tarello. Using Partial Differential Equations to Model TCP Mice and Elephants in Large IP Networks. In *Proc. of IEEE INFOCOM*, 2004.

[MGT99] Vishal Misra, W. Gong, and Don Towsley. Stochastic Differential Equation Modeling and Analysis of TCP Windowsize Behavior. In *Proc. of Performance'99*, 1999.

[MGT00] V. Mishra, W. Gong, and D. Towsley. Fluid based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *Proc. of the ACM SIGCOMM*, 2000.

[MLAW99] J. Mo, R. J. La, V. Anantharam, and J. C. Walrand. Analysis and Comparison of TCP Reno and Vegas. In *Proc. of IEEE INFOCOM*, pages 1556–1563, 1999.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018, Standards Track, October 1996.

[MO99] Archan Misra and Teunis J. Ott. The Window Distribution of Idealized TCP Congestion Avoidance with Variable Packet Loss. In *Proc. of IEEE INFOCOM*, pages 1564–1572, 1999.

[MSZ02] M. Mellia, I. Stoica, and H. Zhang. TCP Model for Short Lived Flows. *IEEE Communications Letters*, 6(2):85 – 88, February 2002.

[MT93] S.P. Meyn and R. Tweedie. *Markov Chains and Stochastic Stability*. Springer, London, 1993.

[MW00] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000.

[MY95] B. Melamed and D. D. Yao. *Frontiers in Queueing: Models, Methods and Problems*, chapter The ASTA Property. CRC Press, 1995. Invited Chapter.

[Neu81] M. F. Neuts. *Matrix Geometric Solution in Stochastic Models*. The Johns Hopkins University Press, Baltimore, MD, 1981.

[New] J. S. Newman. FORTRAN programs for simulation of electrochemical systems. Available at `http://www.cchem.berkeley.edu/~jsngrp`.

[OKM96] T. J. Ott, J. H. B Kemperman, and M. Mathis. The stationary behavior of ideal TCP Congestion Avoidance. Unpublished manuscript, 1996. `http://web.njit.edu/~ott/Papers/Square_Root/TCPwindow.pdf`.

[OLW99] T. J. Ott, T. V. Lakshman, and L. Wong. SRED: Stabilized RED. In *Proc. of IEEE INFOCOM*, pages 1346 – 1355, 1999.

[PAAD03] B. J. Prabhu, E. Altman, K. Avrachenkov, and J. Abadia Dominguez. A Simulation Study of TCP Performance over UMTS. In *Proc. of IEEE VTC 2003-fall*, Orlando, USA, October 2003.

[PF95] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM*, 1998.

[PPP00] R. Pan, B. Prabhakar, and K. Psounis. CHOKe: A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *Proc. of the IEEE INFOCOM*, 2000.

[Pra02] B. J. Prabhu. Energy Efficient Algorithms for Wireless Mobile Networks. M.Sc.(Engg.) thesis, Dept. of Electrical Communication Engg. Indian Institue of Science, January 2002.

[PSAC01] C. E. Price, K. M. Sivalingam, P. Agarwal, and J.-C. Chen. "A Survey of Energy Efficient Network Protocols for Wireless and Mobile Networks. *ACM/Baltzer Journal on Wireless Networks*, 7(4):343–358, 2001.

[PSS01] S. Park, A. Savvides, and M. B. Srivastava. Battery Capacity Measurement and Analysis Using Lithium Coin Cell Battery. In *Proc. of ISLPED*, pages 382–387, 2001.

[Put94] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. J. Wiley and Sons, 1994.

[RB05] G. S. Rajadhyaksha and V. S. Borkar. Transmission Rate Control Over Randomly Varying Channels. *Probability in Engineering and Informational Sciences*, 19(1):73–82, 2005.

[Sch68] P.J. Schweitzer. Perturbation Theory and Finite Markov Chains. *J. Appl. Probability*, 5:410–413, 1968.

[Sch86] P.J. Schweitzer. Perturbation Series Expansion of Nearly Completely-decomposable Markov Chains. In O.J. Boxma, J.W. Cohen, and H.C. Tijms, editors, *Teletraffic Analysis and Computer Performance Evaluation*. Elsevier Science Publishers B.V. (North Holland), 1986.

[Sch93] M. Schäl. Average Optimality in Dynamic Programming with General State Space. *Mathematics of Operations Research*, 18(1):163–172, 1993.

[Sri04] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[SVL02] V. Sharma, J. Virtamo, and P. Lassila. Performance Analysis of the RED Algorithm. *Probability in the Engineering and Informational Sciences*, 16:367–388, 2002.

[SW93] S. Stidham and R. Weber. A survey of Markov decision models for control of networks of queues. *Queueing Systems*, 13:291–314, 1993.

[TM04] T. A. Trinh and S. Molnar. A Comprehensive Performance Analysis of Random Early Detection Mechanism. *Telecommunication Systems*, 25:9–31, January 2004.

[TSG99] TSG-RAN. Common Test Environment for User Equipment (UE) Conformance Testing. Technical Report TS 34.108, version 4.1.0, 3GPP, 1999.

[UMT98] UMTS 30.03 version 3.2.0. Selection procedures for the choice of radio transmission technologies of the UMTS. Technical Report TR 101 112, ETSI, April 1998.

[VBB00] M. Vojnovic, J. Y. Le Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times. In *Proceedings of IEEE INFOCOM'2000*, pages 1303–1312, Tel Aviv, Israel, March 2000.

[WM95] H. S. Wang and N. Moayeri. Finite-state Markov channel-a useful model for radio communication channels. *IEEE Trans. Vehicular Tech.*, 44(1):163–171, 1995.

[Wol89] R. Wolff. *Stochastic Modeling and the Theory of Queues.* Prentice-Hall, Englewood Cliff, NJ, 1989.

[XHR04] L. Xu, K. Harfoush, and I. Rhee. Binary Increase Congestion Control (BIC) for Fast Long-Distance Network. In *Proc. of the IEEE INFOCOM*, March 2004.

[YHE02] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the IEEE INFOCOM*, pages 1567–1576, 2002.

[ZC90] L. Zhang and D. Clark. Oscillating Behavior of Network Traffic: a Case Study Simulation. *Internetworking:research and experience*, 1(2):101–112, 1990.

[ZCR00] M. Zorzi, A. Chockalingam, and R. Rao. Throughput Analysis of TCP on Channels with Memory. *IEEE Jl on Selected Areas in Comm.*, 18(7):1289–1300, July 2000.

[ZR01] Michele Zorzi and Ramesh R. Rao. Energy Efficiency of TCP in a Local Wireless Environment. *Mob. Netw. Appl.*, 6(3):265–278, 2001.

[ZVW82] Z.Rosberg, P. P. Varaiya, and J. C. Walrand. Optimal Control of Service in Tandem Queues. *IEEE Transactions on Automatic Control*, AC-27(3):600–610, June 1982.

Résumé

Cette thèse contient quelques applications des chaînes de Markov et des processus de décision markoviens pour le contrôle de congestion et de puissance. D'abord nous étudions le comportement de la taille de la fenêtre d'une source qui utilise l'algorithme MIMD. Nous montrons que le logarithme de la taille de la fenêtre suit une récurrence stochastique additive, et est une chaîne de Markov. Nous obtenons la distribution et les moments de la taille de la fenêtre lorsque le processus de perte de paquet est indépendant de la taille de la fenêtre. Nous montrons aussi que le débit obtenu par une source est proportionnel à l'inverse de la probabilité de perte d'un paquet. Ensuite, nous analysons le processus de la taille de la fenêtre d'un algorithme de contrôle de congestion en temps continu. Nous obtenons l'équation de Kolmogorov pour la distribution de ce processus en état stationnaire, et montrons que ce processus est équivalent au processus de temps d'attente dans une file d'attente où le taux d'arrivée et de service dépend du processus de temps d'attente. Suite à cette équivalence, nous pourvoyons des conditions sous lesquelles deux algorithmes ont le même comportement. Puis, nous étudions le processus de rapport de deux sources qui utilisent l'algorithme MIMD et qui se partagent la capacité d'un goulot d'étranglement. Pour les sources homogènes, nous montrons que l'indice d'équité s'améliore si le processus de perte de paquet dépend du débit. Pour les sources hétérogènes, nous montrons que l'intensité du processus de perte de paquet doit être supérieure à une constante qui dépend des paramètres des algorithmes pour que l'indice d'équité s'améliore. Lorsque le processus de perte de paquet est synchronisé et plusieurs sources utilisant soit l'algorithme AIMD soit l'algorithme MIMD se partagent une liaison, nous observons que les sources utilisant l'algorithme AIMD obtiennent un débit indépendant de la capacité de la liaison. Ensuite, nous présentons un modèle stochastique pour obtenir la distribution jointe du nombre instantané de paquets et de sa moyenne mobile dans une file d'attente RED. Lorsque le paramètre du calcul de la moyenne mobile tend vers zéro, nous utilisons la théorie de perturbation singulière afin d'obtenir la distribution conjointe. Nous montrons que le nombre moyen peut être calculé à partir d'une équation à point fixe. Ensuite, nous étudions un problème de commande optimale en temps discret. Un appareil mobile veut transmettre des paquets et conserver son énergie en même temps. Nous montrons que la politique optimale est un contrôle à seuil. Enfin, par simulations, nous étudions le délai des flots TCP sur la voie descendante de l'UMTS lorsque deux politiques différentes de commutation de canaux sont utilisées.

Mots-clés — Modélisation stochastique, chaînes de Markov, processus de décision markoviens, algorithmes de contrôle de congestion, MIMD, équité, file d'attente RED, séparation d'échelle de temps, UMTS, politiques de commutation de canaux.

Abstract

This thesis is based on some applications of Markov chains and decision processes for performance analysis of congestion avoidance algorithms and power control. First, we study the behaviour of the window size of an MIMD congestion control algorithm when subject to different loss processes. We show that the logarithm of the embedded window size follows an additive recursive equation, which can be modelled as a Markov chain. Through this model, we obtain the distribution of the window size and its moments when losses are independent of the window size. We then propose an approximation for the loss process in which each packet is lost with a constant probability, and show that the throughput is inversely proportional to the packet loss probability. Next, we study a continuous time model for the window size of a general increase and instantaneous decrease congestion control algorithm. We show that the window size is a Markov process for which we give the Kolmogorov equation for the stationary distribution function. By applying a transformation, we also show that the window size is equivalent to the workload in a queue with state-dependent arrival and service rates. Through the Kolmogorov equation and the analogy with a queueing system we provide conditions under which two congestion control algorithms have related window size behaviour. For example, it is shown that the behaviour of the window size of the MIMD algorithm with a linear loss is similar to that of the window size of the AIMD algorithm with a constant loss intensity. Next, we model the instantaneous ratio of the sending rates of two competing MIMD sessions as a Markov chain. Through this model we study the fairness properties of the MIMD algorithm when subject to different loss processes. For heterogeneous sources, we show that the fairness index can be improved by introducing rate-dependent losses at an intensity which is greater than a certain threshold intensity. We also study the bandwidth sharing between AIMD and MIMD sources when the losses are synchronous. The results indicate that the AIMD sources obtain a rate that is independent of the link capacity whereas the MIMD sources utilise the rest of the capacity. Next, we model the instantaneous and the average queue sizes of a RED enabled queue as a non-homogeneous Quasi-Birth-Death process. In the limit when the averaging parameter goes to zero, we use the singular perturbation technique to find the joint distribution of the average and the instantaneous queue sizes. In this limiting regime, the expected average queue size can be approximated by the solution to a fixed point equation. A problem related to energy-delay tradeoff in a wireless device is studied next. The device has a finite energy battery using which it wants to transmit data. The battery can recover some charge when left idle. In each slot, the device has to decide whether to transmit data or to leave the battery idle in order to increase the battery lifetime. We formulate this problem as a Markov decision process and provide conditions under which the optimal policy is of threshold type. Finally, through simulations we study the effect of two threshold based channel switching policies in UMTS on the expected delay performance of TCP sessions.

Keywords — Stochastic modelling, Markov chains, Markov decision processes, congestion control algorithms, MIMD, fairness, RED queue, time-scale seperation, UMTS, channel switching policies.