Routing and Scheduling Problems in Data Networks

Gabriel Scalosub

Technion - Computer Science Department - Ph.D. Thesis PHD-2007-14 - 2007

Routing and Scheduling Problems in Data Networks

Research Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Gabriel Scalosub

Submitted to the Senate of the
Technion - Israel Institute of Technology

Heshvan 5768 Haifa October 2007

The research thesis was done under the supervision of Prof. Seffi Naor, Assoc. Prof. Danny Raz, and Dr. Adi Rosén in the Department of Computer Science.

I would like to thank my advisors, which have assisted me and guided me throughout the long stretches of my work. You have been a model and an inspiration to me during these past years, and I cherish my good fortune of having the chance to have you as my advisors. Seffi, for the great encouragement and support you've given me in pursuing new problems and new ideas, for great advice and for pointing the right way when at times the horizon seemed a little foggy. Adi, for the endless hours together over many coffee cups, spent stretching ideas and discussing life in general, and for always demanding that my results be precise and elegant. Danny, for the pleasure of working with you, for the constant care and support you've given me, and for having your door always open. It was always reassuring to know I have you to bounce off thoughts and ideas.

I would like to thank David Hay, Dudu Amzallag and Reuven Bar Yehuda, with whom I had the pleasure to work.

I would like to thank my many friends at the Technion, and outside, which have been a constant source of support and help.

I would like to thank my mother and brother who where always there for me, constantly looking for ways to assist me and give me the time and possibility to focus on my work. Your love and support have been, and will always be, invaluable.

Finally, I would like to thank my better half, Dina, for putting up with my absent-mindedness, the pressure, and the intensity of it all. This couldn't have happened if not for your endless love and encouragement.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Figures

List of Tables

Abstract

Data communication networks have a major effect on many aspects of modern day life. The complexity of these networks, both in terms of architecture, and in terms of the services these network are required to provide, necessitate the design of novel algorithms and protocols that can deliver higher throughput rates and guarantee the needed Quality-of-Service. In this work we study several fundamental problems arising in modern networks, concentrating on real-life scenarios where limitations are imposed on solving the problem, either in terms of lack of information regarding future events or in terms of lack of coordination between the various system's components. We take on a *competitive approach* in the design and analysis of our proposed algorithms, which enables us to provide worst-case performance guarantees compared to the optimal performance. This makes our algorithms globally applicable, independent of specific traffic patterns or underlying traffic distributions.

We first consider the problem of bufferless routing and scheduling in optical networks, and present the first online algorithms for the problem. Our algorithms apply to the case where the underlying topology is a line or a ring. We give guarantees as to the worst-case performance of our algorithms, and present matching lower bounds.

Next we consider the problem of guaranteeing low jitter for multiple streams using a shared buffer. We show that while the problem can be solved optimally in an offline setting, in the online setting a substantial resource augmentation is required in order to provide optimal jitter. The amount of resource augmentation depends linearly on the number of streams, thus our results indicate that jitter regulation does not scale well when the number of streams increases.

We then consider the problem of information gathering on the line, with limited buffer space at each node. We extend the common model for such problems by considering the adversary's rate. This enables us to provide better guarantees which depend not only on the network size, but also on the adversary's rate and the amount of buffer space available at the nodes. Our results can also guide better buffer provisioning in some cases which can guarantee optimal throughput.

Lastly, we consider the problem of multiple access in wireless networks with soft collisions. In this model collisions may occur due to proximity between transmitting nodes, but not every simultaneous transmission results in a collision. We take on a game theoretic approach and show that for the case where interferences are homogeneous, selfish behavior may result in an exponential degradation in throughput. However, there exists a penalizing scheme on the selfish agents which can guarantee at least a $1/e$ fraction of the optimal throughput. Based on these techniques, we develop a fully distributed protocol for non-homogeneous interferences, which outperforms natural protocols for the problem. We also show using simulation that this protocol is very robust in the sense that it performs well under varying load situations.

Abbreviations and Notations

α	The maximum ratio between any two packets' lengths	9
R	The number of different packet lengths	9
n	The number of nodes in the network	9
β	The maximum ratio between any two packets' weight-to-length ratios	10
A	The set of packets scheduled by the online algorithm	15
O	The set of packets scheduled by an optimal schedule	15
N	The set of packets never scheduled by the online algorithm	15
ϕ	The golden ratio	18
M	The number of streams	23
X	The inter-release time of packets by the source	23
B	The available buffer size (at every node)	24
σ	The sequence of packets	25
s	The release schedule	25
λ	The link capacity	25
$J^{\sigma}(s)$	The jitter obtained by schedule s given sequence σ	25
$\mathrm{MJ}^{\sigma}(s)$	The max-jitter obtained by schedule s given sequence σ	25
r	The rate of the adversary	41
$\overline{R}$	The profile of strategies chosen by the set of agents	61
$\alpha_{i,j}$	The interference agent j inflicts on agent i when both transmit	61
$\varphi(\overline{R})$	The social welfare value given profile $\overline{R}$	62
$U_i(\overline{R})$	The utility of agent i given profile $\overline{R}$	62
v_k	The social welfare value when exactly k agents transmit	64
$\mathrm{PoA}^{(n)}$	The price of anarchy given n agents	67
$\mathrm{PoS}^{(n)}$	The price of stability given n agents	67

Chapter 1

Introduction

1.1 Future Networking Environments

Data communication networks are becoming an integral part of our every day modern life. The increasing influence these networks have on our lives, and the constant increase in the number of applications which make use of these networks, have motivated a vast amount of research in the past decades, attempting to improve their performance, in order to make them more suitable to provide new abilities, and facilitate new services.

The many challenges tackled in recent years include the need for providing Quality-of-Service (QoS) guarantees. This was greatly driven by the popularity of multimedia applications that are based on audio and video streaming, as well as the continuing increase in traffic volume, and the deployment of optical devices in core networks.

In real life networks, it is often the case that the input is not given to the system in advance. In most cases the input to the system is revealed over time, and the system only has information regarding present and past events. The various mechanisms in the system have no information on future input with which they will be confronted at a later time. Typical examples include new users joining the network, new streams of information traversing the network, or new packets arriving at a buffer. Such a setting is usually referred to as an *online setting*, and it is usually the common case in the various issues arising in the operation of modern networks. The difficulty in these settings is mostly due to *lack of information*. All the problems we consider can be also cast in an *offline setting*, where the system has full knowledge of the entire input. The difficulty then becomes mostly *computational*, where it might not always be possible to provide an efficient algorithm that is guaranteed to provide an optimal solution.

In some networking scenarios, entities in the same network belong to different organizations and have different business objectives. This is the case, for example, with autonomous systems in the Internet, or base stations in wireless networks. Some of the problems addressed in this work investigate the performance of such systems, by modeling them as a game between various independent agents. It is customary to assume that the agents are *selfish*, and are trying to optimize some private utility regardless of the effect their choices might have on the overall performance of the system. In these cases, the research goal is to evaluate the degradation in performance resulting from selfish behavior, and to design mechanisms which may improve the overall system's performance. The main difficulty in guaranteeing good performance in such

cases is due to *lack of coordination* between the various agents. The goal of the above mentioned mechanisms is to inflict a certain amount of coordination upon the various agents which are part of the network, so as to obtain better performance.

1.2 Problem Modeling and Criteria for Evaluating Performance

We formulate the problems at hand as optimization problems, where the input may consist of the size of the network, the number of data bits to be transmitted, the available network resources, etc. The goal is to optimize a certain global function which represents the system's performance, under the various restrictions posed by the specific problem we consider.

The first element comprising our optimization problems is a set of constraints, which implicitly define a collection of feasible solutions. In most of the problems we consider, these constraints are derived from several types of requirements. One such type are Quality-of-Service requirements, such as maximum allowable delay, specific deadlines, maximum allowable jitter, etc. (See e.g., chapters 2 and 3). Another type of requirements are due to the limited nature of the network resources, such as bounded buffer space, limited capacity links, etc. (See e.g., chapters 3 and 4). Another type of requirements are due to the network-model underlying the problem, which may include interferences between network elements, and other specific network characteristics (See e.g., chapters 2 and 5).

The second element of each optimization problem is the global function which our algorithms aim at optimizing. In most cases, the network's resources cannot support the overall amount of traffic, or it simply cannot satisfy all the different demands. In such cases, a choice must be made by the system as to which traffic to eliminate, or alternatively, which demands to satisfy. A judicious choice in such cases can make the difference between a network providing good overall performance, even in cases of overload, and networks with poor performance. These choices are therefore reflected in the overall system's *throughput*, which should be maximized (See e.g., chapters 2, 4, 5).

In some of the problems addressed in this work, the network should be able to handle all the traffic, such that the minimum amount of obstructions are inflicted. Such obstructions may be the delay in handling the traffic, the jitter, or the packet loss. The goal in such cases would be to minimize the global function which captures the system's performance for the problem at hand (See e.g., Chapter 3).

1.3 Algorithmic Concepts

In this thesis we study different algorithmic aspects of optimization problems related to communication networks. Our algorithms are expected to be deployed by different components of the network, performing different algorithmic tasks. We mainly focus on the following algorithmic aspects:

Admission Control. When the network resources are limited, either in terms of time (e.g., the existing tasks already use all the available time), buffer space, or other network capacities, then accepting further obligations cannot be done recklessly. One should carefully consider which

further tasks to accept, and which to deny. Admission control is specifically important under high-load conditions, where there are many requests contending for the system's resources. In such a case the admission control mechanism needs to take into consideration both the revenue accrued from accepting a request, as well as the impact such an acceptance might have on the overall system's resources.

Scheduling. In most communication networks, after employing an admission control mechanism, the existing tasks must be scheduled in order to optimize a global function (e.g., jitter, or delay), or simply in order to guarantee a feasible solution within the boundaries imposed by the system's constraints. Such scheduling procedures are at the heart of many communication systems, e.g., switching mechanisms, access control, etc.

Distributed and Local Control Mechanisms. Small networks are usually easy to handle. One can propagate information to a central entity which makes choices and decisions based upon the global state of the system, and forwards the decisions back to the network elements. However, as networks become larger, implementing such central mechanisms becomes harder and harder, both due to the high overhead of transferring data to the central mechanism, as well as to the prohibitive amount of time such a procedure may require sometimes. In many of the problems we consider, we aim at devising distributed and local control mechanisms, which are on the one hand robust and may be applied to systems of variable sizes, and on the other hand have good performance characteristics, and can still exhibit good behavior compared to centralized mechanisms with access to global information.

1.4 Analysis Concepts

The analysis concepts underlying our work are derived from the nature of the problems we consider. We analyze the performance of our online algorithms using *competitive analysis* (see [21, 67]). The quality of the output produced by our algorithm is compared to the optimal output possible, which can be viewed as the result of a clairvoyant algorithm with unbounded computational power. This approach is robust in the sense that it makes no assumptions on the input to the problem, and therefore any guarantee provided by such an algorithm would be globally applicable. That is, we do not need to use any assumption regarding the input, any stochastic behavior, or any underlying distribution of the input. In real life, such a characterization of traffic is hard to come by, and even at times where such estimates are at hand it is not always clear how to harness this information into an algorithm which provides a better performance.

A deterministic online algorithm for a maximization problem is said to be δ-*competitive*, for some $\delta \geq 1$, if for any input sequence σ,

$$A(\sigma) \geq \frac{1}{\delta} O(\sigma) + b.$$

Here $A(\sigma)$ and $O(\sigma)$ represent the value obtained by the solution produced by the online algorithm and an optimal solution, respectively, and b is some constant independent of σ.

In a game theoretic setting, where we have selfish agents trying to optimize some personal utility function, we focus our attention on the performance of the system in Nash Equilibrium,

where no agent can do better by altering its chosen strategy (See e.g., [62]), and compare this performance to the optimal system's performance. Further details in this respect appear in Chapter 5.

1.5 Overview of the Thesis

This thesis studies several fundamental problems arising in optimizing the performance of future networks. In what follows we give an overview of our results in each of these domains.

Online Time-Constrained Scheduling in Line and Ring Networks. We consider the problem of scheduling a sequence of packets over a linear network, where every packet has a source and a target, as well as a release time and a deadline by which it must arrive at its target. The model we consider is bufferless, where packets are not allowed to be buffered in nodes along their paths other than at their source. This model applies to optical networks where opto-electronic conversion is costly, and packets mostly travel through bufferless hops.

The offline version of this problem was previously studied in [2]. We focus our attention on the online version of the problem, where we are required to schedule the packets without knowledge of future packet arrivals. We present the first online algorithms for several versions of the problem. For the problem of *throughput maximization*, where all packets have uniform weights, we give an algorithm with a logarithmic competitive ratio, and present several lower bounds. For other weight functions, we show algorithms that achieve optimal competitive ratios. These results also appear in [61].

Jitter Regulation for Multiple Streams in Capacitated Networks. For widely-used interactive communication, it is essential that traffic is kept as smooth as possible; the smoothness of the traffic is typically captured by its *delay jitter*, i.e., the difference between the maximal and minimal end-to-end delays. The task of minimizing the jitter is done by jitter regulators that use a limited-size buffer in order to shape the traffic. In many real-life situations regulators must handle multiple streams simultaneously and provide low jitter for each of them separately. Moreover, communication links have limited capacity, and these may pose further restrictions on the choices made by the regulator. In this work we investigate the problem of minimizing jitter in such an environment, using a fixed-size buffer.

We show that the offline version of the problem can be solved in polynomial time by introducing a polynomial time offline algorithm that finds a release schedule with optimal jitter. When regulating multiple streams in the online setting, we note that previous results in [56] can be extended to an online algorithm that uses a buffer of a larger size, while obtaining the optimal jitter possible with the original size buffer. The buffer size increase is proportional to the product of the original buffer size and the number of streams. The question arises whether such a resource augmentation is essential. We answer this question in the affirmative by proving a lower bound that is tight up to a factor of 2, thus showing that jitter regulation does not scale well as the number of streams increases unless the buffer is sized-up proportionally. Part of these results also appear in [36].

Rate vs. Buffer Size: Greedy Information Gathering on the Line. We consider packet networks with limited buffer space at the nodes, and study the question of maximizing the number of packets that arrive to their destination rather than being dropped due to full buffers. We initiate a more refined analysis of the throughput competitive ratio of admission and scheduling policies in the Competitive Network Throughput model [5], taking into account not only the network size but also the buffer size and the injection rate of the traffic.

We specifically consider the problem of information gathering on the line, with limited buffer space, under adversarial traffic. We examine how the buffer size and the injection rate of the traffic affect the performance of the greedy protocol for this problem. We establish upper bounds on the competitive ratio of the greedy protocol in terms of the network size, the buffer size, and the adversary's rate, and present lower bounds which are tight up to constant factors. These results show, for example, that provisioning the network with sufficiently large buffers may substantially improve the performance of the greedy protocol in some cases, whereas for some high-rate adversaries, using larger buffers does not have any effect on the competitive ratio of the protocol. These results also appear in [65].

Soft Collisions in Wireless Networks. In many modern wireless scenarios, the distinction between collisions and interferences is not always clear. The main concern is the ability of the receiving stations to distinguish between the signal of the message and the noise created by other signals in the same proximity. We concentrate on a wireless environment where fixed stations need to communicate with clients in their vicinity, over a common communication channel (e.g., a single radio mesh network), and focus on a distributed setting where there is no central entity managing the various transmissions. In such systems, unlike other multiple access environments, several transmissions may succeed simultaneously, depending on spatial interferences between the different stations.

We use a game theoretic view to model the problem, where the stations are selfish agents which aim at maximizing their success probability. We show that when interferences are homogeneous, the system's performance necessarily suffers an exponential degradation of performance in an equilibrium, due to the selfishness of the stations. However, when using a proper penalization scheme for aggressive stations, we can ensure the system's performance value is at least $1/e$ of the optimal value, while still being at equilibrium.

Based upon this study, we develop a simple distributed local heuristic for the problem, and show using extensive simulations that this heuristic achieves a very good utilization of the radio resources compared to other commonly used protocols. Moreover, this local heuristic is very robust and performs well under different load conditions. These results also appear in [60].

Chapter 2

Online Time-Constrained Scheduling in Line and Ring Networks

2.1 Introduction

As technology advances, communication networks are constantly going through rapid change. The classic best-effort mechanisms are given up in favor of networks that are able to provide Quality-of-Service (QoS) guarantees. The growing use of multimedia applications motivate this transition. Such applications involve continuous transmission of data, which requires some guarantees as to its arrival time, bandwidth allocation etc. [51].

It is often the case that the overall number of packets destined to be transmitted through the network exceeds the network's capacity. In such cases, packets are either delayed or dropped. When considering streaming video or audio data, there is very little point in delaying such packets more than some predetermined period of time. Take, for example, a home user listening to the radio over the Internet. We can model such a transmission by considering every packet to have a certain deadline by which it must arrive at its destination. In such a setting, having the packet arrive after its deadline is of no use.

Real life applications vary in importance and value as well, thus rendering some packets more important than others. Consider, for example, the case of MPEG encoding, where some packets are more important than others when reconstructing the image at the target. This situation makes it vital to decide which packets to schedule at any given time, such that the decision will eventually result in a "best" possible set of packets, which are all delivered by their deadline.

When considering such packets with their corresponding deadlines, one would want to take into account both the packet's importance as well as its deadline when trying to determine which packet to route first. Additionally, packets can have different values, according to the end user's willingness to pay for an improved quality of service. In such a scenario, delivering valuable packets on time would mean more profit for the service provider, which should naturally be maximized. Time-constrained traffic is also the common case in real-time applications, such as avionics, industrial process control, and automated manufacturing, which necessitate coping with time constrained communication in interconnection networks [64].

In this chapter we consider the problem of online scheduling a sequence of packets, each

with a deadline constraint. The model underlying our work is a *bufferless* scheduling environment. In this model, a packet can only be stored at its source, and cannot be buffered in any node along its path. Once a packet has left its source, it must move along its designated path without interruptions or delays, until arriving at its destination. Any interruption or delay causes the packet to be dropped. This model is the common setting in optical networks, where trying to buffer packets in nodes along the path requires opto-electronic conversion of the signal, a prohibitively costly operation. This is the case in Wavelength Division Multiplexing (WDM) networks, where a packet is assigned a wavelength along which it is supposed to be transmitted throughout its path.

We restrict our attention to specific network topologies such as the line and the ring. The results of [1] motivate this focus, since under common complexity assumptions, for arbitrary graphs, no reasonable approximation can be obtained in polynomial time. Moreover, focusing on simple network topologies like the line topology or the ring topology is motivated by considering electro-optical interconnection networks. In such networks, we might have a packet's path go through several long bufferless hops with very few nexus points, each enabling the expensive optical-electric conversion. This occurs for example in a mesh network topology, employing a dimension-order routing policy. In such a case we can use a bufferless strategy along rows and columns, and perform a conversion to change dimensions (see [64]). Another advantage in considering simple topologies is the fact that they usually adhere to simple routing-path selection. In cases of less regularly structured networks, it is often the case that packets are routed along subnetworks of such simple topologies.

2.1.1 Our Results

We present the first online algorithm for bufferless scheduling of packets with deadline constraints in a linear network topology. Our goal is to maximize the total weight of packets delivered by their deadlines. A packet p contributes its weight to the overall weight gained by the algorithm only if it arrives at its target node by its deadline. We can further show that these results extend to a ring network topology.

We present results for several special cases of the problem, determined by the weights given to the packets. In the *Throughput Maximization* problem the packets have uniform weights, i.e., for every packet p, its weight is equal to some constant w, where without loss of generality $w = 1$, and thus our goal is to maximize the number of packets scheduled successfully. In the *Maximum Network Utilization* problem the weight of each packet is defined to be its path length. The optimization problem in this case can be considered as trying to maximize the utilization of the network over time, where only packets scheduled successfully contribute to the network utilization. We further present results for the general case of arbitrary weights.

In Section 2.2 we present an $O(\min\{\log \alpha, R\})$-competitive algorithm for the throughput maximization problem, where α is the ratio between the length of the longest path of a packet in the input sequence and the length of the shortest path, and R is the number of different path lengths appearing in the sequence. This reduces to an $O(\log n)$-competitive algorithm in the worst setting, where n is the number of nodes in the network. Unlike the results of [46] and [33] for task scheduling on a single machine, our algorithm need not know the value of the parameter

α beforehand. We give an example exhibiting our analysis to be tight up to a constant factor. We additionally show that no deterministic algorithm for the problem can achieve a competitive ratio better than 2.

In Section 2.3 we give a constant competitive algorithm for the problem of maximizing network utilization. This algorithm is an adaptation to our model of an algorithm given in [32]. We further derive an $O(\beta)$-competitive algorithm for arbitrary weights where β is the maximum ratio between any two packets' weight-to-length ratio. Due to the results of [19], this is the best possible, up to a constant factor.

In Section 2.4 we show how our results can be applied to a ring network topology.

2.1.2 Previous Work

The offline version of our problem in the linear network topology was first considered by Adler *et al.* in [2]. They restricted their attention to the problem of throughput maximization and showed that it is NP-hard, and further provided a 2-approximation algorithm for the problem. Another model considered in [2] is the *buffered* model, where packets are allowed to be stored in a buffer of any node along their path. Adler *et al.* showed that allowing the packets to be buffered along their paths can increase the throughput by at most an $O(\log \gamma)$ factor, relative to the throughput obtained by a bufferless schedule, where γ is the minimum among the network size, the number of packets in the instance and the maximum slack a packet has.[1] Adler *et al.* devised a distributed online algorithm for the buffered case, which mimics the approximation algorithm given for the buffereless case. An extension of these results was later given by Adler *et al.* in [1], where they present algorithms for several versions of the time constrained scheduling problem, all in an offline setting. They first describe a 2-approximation algorithm for the bufferless case in a linear network, where packets are allowed to have arbitrary weights. They further consider the case where the underlying network topology is a tree or a mesh in the bufferless setting. For this problem they present constant-approximation algorithms for both the throughput maximization problem as well as for arbitrary weights. For the buffered case in the tree and mesh topologies, they devise an algorithm based on the algorithm for the bufferless case, with a logarithmic approximation guarantee.

The hardness results appearing in [1] motivate the focus on particular network topologies as they show that for any $\varepsilon > 0$, there is no $k^{1-\varepsilon}$-approximation algorithm for the problem in general networks, unless NP=ZPP, where k is the number of packets in the instance. This hardness result is based on the hardness of MAX-INDEPENDENT-SET, and it holds even if the underlying topology is either a directed acyclic graph or a planar graph.

The only result regarding the online version of the problem is given in [1], where they show that no deterministic online algorithm can achieve a competitive ratio better than $\Omega(\log n)$ when the underlying graph is a tree, in both the bufferless and the buffered settings, where n denotes the size of the network. One can compare this result with our upper bound for the linear network topology, which is guaranteed to be $O(\log n)$-competitive.

Our problem is closely related to interval scheduling problems and other call control models, e.g., [32], [52], and [3]. In the online interval scheduling problem we are given a sequence of

[1] For the definition of slack, see Section 2.1.3.

intervals to schedule on a line segment. In some cases the problem can be solved in polynomial time, e.g., the case where the intervals are given in non-decreasing order of their left end-point, all having uniform weights, and preemption is allowed. In other cases however there are lower bounds on the attainable competitive ratio of any online algorithm, e.g., the case where the weight of an interval is defined to be its length, even in a randomized setting [52], and the case where intervals have uniform weights in a deterministic setting [32]. These lower bounds apply to non-preemptive scheduling of the intervals. Our model however is not reducible in the general case to either of these. The main difference between our model and the ones mentioned above is the concept of time, which introduces further constraints on the scheduling problem. Further results related to our problem involve multiple bin-packing, dealt with in the context of call admission control and wavelength division multiplexing in optical networks [14], which were later adapted to the case where calls are allowed to be preempted [3].

Some results regarding online task scheduling on a single machine, where each job must terminate by a certain deadline, are also related to our problem. Baruah *et al.* show in [19] that when tasks may have arbitrary weights, no deterministic online algorithm can achieve a competitive ratio better than $\Omega(\beta)$, where β is the ratio between the largest and the smallest weight-to-processing-time ratio of the packets in the instance. In [46] Koren and Shasha present an online algorithm for the problem, whose guarantee is exactly that of the lower bound in [19]. Their algorithm need know the value of β in advance. A guarantee based on a different parameter is given by Garay *et al.* in [33] for the problem of throughput maximization. They present an algorithm that is guaranteed to be $O(1/\kappa)$-competitive, where κ is the minimum ratio between the slack and the processing time of all jobs in the request sequence. In this case as well, the algorithm has to be given the value of κ in advance.

2.1.3 Model

Our main results will be described for the linear network. We model our problem by a digraph $G = (V, E)$, where $V = \{1, \ldots, n\}$, and $E = \{(i, i+1) | 1 \leq i \leq n-1\}$. An instance comprises additionally of a set of packets that are to be routed through the network. Each packet p is specified by a tuple $(s_p, t_p, r_p, d_p, w_p)$, where s_p and t_p denote the source and target nodes respectively, r_p is the packet's release time, i.e., the time at which the packet is available for routing, d_p denotes the packet's deadline, and w_p is the packet's weight. We denote by $|p| = t_p - s_p$ the *length* of packet p. The algorithm learns of packet p in time r_p. The above definitions make it natural to consider the *slack* each packet has, also known as *laxity*, defined by $\ell(p) = d_p - r_p - |p|$. The slack of packet p captures the notion of the maximum amount of time a packet can wait at its source node if it is to arrive at its target node by its deadline. We denote by $\ell_t(p) = d_p - t - |p|$ the *residual slack* of packet p in time t. A packet can be scheduled to leave its source at any time t for which $\ell_t(p) \geq 0$. We consider a *synchronous* model, where at each time step at most one packet can be transmitted on any edge, and we focus our attention on the *bufferless* case. We make no restriction on the amount of storage available at any node. We further assume packets can be *preempted* but cannot be *rescheduled*. Preemption means that a packet on route to its destination can be stopped, in which case it is dropped and cannot be rescheduled, even if its residual slack allows it. Every packet arriving at its destination

by its deadline contributes its weight to the overall weight obtained, and is considered successfully scheduled. Every other packet contributes 0 to the overall obtained weight. The goal is to maximize the weight obtained.

In what follows we will use the following notation. Let $M = \max_p |p|$ and let $m = \min_p |p|$. We let α denote the ratio M/m and R is the number of different packet lengths appearing in the input. Define the *density* of packet p to be $\rho(p) = w_p/|p|$. Denote by $\rho_{\min} = \min_p \rho(p)$, $\rho_{\max} = \max_p \rho(p)$ and let $\beta = \rho_{\max}/\rho_{\min}$.

Geometric Intuition We follow the geometric representation introduced in [2]. We define the concept of *waves* upon which we "mount" the packets to be scheduled. Consider a two dimensional array whose X-axis represents the linear network, numbered $1, \ldots, n$ to designate the network nodes, and its Y-axis represents time, numbered $1, 2, \ldots$ to designate discrete time steps. Given a packet p that was presented at time r_p with slack $\ell(p)$, in order for it to arrive at its destination by its deadline, it must be sent from its source at some time $t \in \{r_p, \ldots, r_p + \ell(p)\}$. Every such scheduling of p starting at t can be geometrically viewed as packing an interval of length $|p|$ on a SW-NE line starting at point (s_p, t) and ending in $(t_p, t + |p|)$. We call each such SW-NE line a *wave*. Every such wave represents the network resources used over time. Each packet has a set of *eligible* waves, defined according to the packet's parameters, where a packet can be mounted on any of its eligible waves. Figure 2.1 shows an example of the waves eligible for a packet p for which $\ell(p) = 4$, and the location in which it can be mounted in every one of them. For each packet p, we consider the waves eligible for packet p as ordered from earliest (crossing point (s_p, r_p)) to latest (crossing point $(s_p, r_p + \ell(p))$). A feasible schedule solution is a packing of the packets upon the waves, such that on any wave no two packets intersect, and every packet is scheduled on at most one wave.

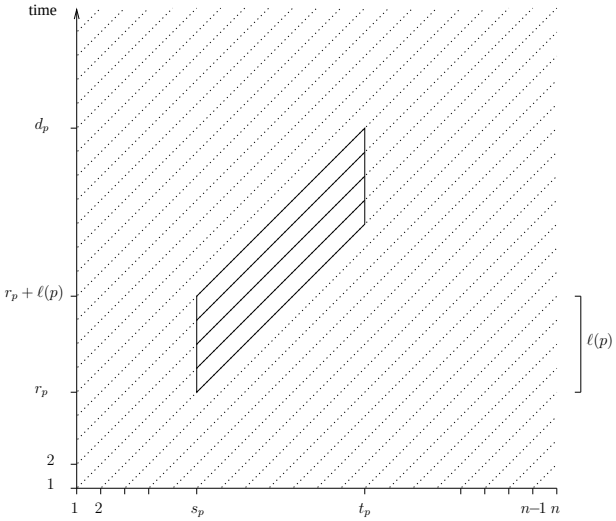


Figure 2.1: Geometric representation of waves

2.2 Throughput Maximization

We first consider the case where for every packet p, $w_p = w$ for some constant w. Without loss of generality we assume $w = 1$, and thus our goal is to maximize the number of packets scheduled.

2.2.1 The Case of Zero Slack

Consider first the throughput maximization problem where all packets have zero slack. Using our geometric intuition, in this case, every packet has only one eligible wave. We therefore seek to compute a maximum-independent set, in an online fashion, for each wave independently. Since preemption is allowed, for such instances this can be done optimally. To see this notice that when focusing on a single wave, the packets corresponding to this wave are given in increasing order of their left end-point. This is due to the fact that packet p is introduced in time r_p. We can therefore preempt a currently scheduled packet q on the wave in favor of a packet p for which $t_p < t_q$. This mimics exactly the behavior of an offline algorithm for finding a maximum independent set in an interval graph in these settings, which finds an optimal solution. If we allow packets to have positive slack, the plot thickens, as demonstrated in the following section.

2.2.2 Online Bufferless Lower Bound

Theorem 2.2.1. *No deterministic online algorithm can achieve a competitive-ratio better than* 2*. This holds even if rescheduling is allowed.*

Proof. Consider a linear network with 4 nodes $\{v_1, \ldots, v_4\}$. We now describe an adversary. The adversary releases at time 0 a packet p with slack $= 1$, going from v_1 to v_4. If the algorithm schedules it on its first wave (i.e., it starts moving at $t = 0$), then the adversary releases at time $t = 2$ a packet q with zero slack, going from v_3 to v_4. If, on the other hand, the algorithm schedules p on its second wave (i.e., it starts moving at $t = 1$), then the adversary releases at time $t = 2$ a packet q with zero slack, going from v_2 to v_3. In either case, the algorithm can deliver at most one of the two packets, while an optimal solution delivers both. Notice that in both cases, if the algorithm preempts p in favor of q, then it cannot reschedule p on any other wave, because at the time of preemption p has a negative residual slack, i.e., it can no longer reach its target node by its deadline. We can repeat this procedure an arbitrary number of times, thus ensuring no deterministic online algorithm can achieve a competitive ratio better than 2. □

2.2.3 Online Bufferless Upper Bound

A Simple Randomized Strategy

A simple greedy strategy can be used to devise a randomized $O(\log n)$-competitive non-preemptive algorithm for the problem. Consider a new packet just arrived. If it can be scheduled (considering the previously scheduled packets) on any wave, then schedule it. Otherwise, discard it. Since this algorithm is $(\alpha + 1)$-competitive when considered on any single wave, using the multiple-bin packing methodology appearing in [14], it follows that the above algorithm is

$(\alpha + 2)$-competitive for our problem. We now introduce randomization: Consider a partition of the packets into $O(\log n)$ classes according to their length, where class i consists of all packets whose length falls in the interval $(2^i, 2^{i+1}]$, and we have $i = 0, \ldots, \log n - 1$. Pick uniformly at random a class i, and use the greedy strategy described above to schedule only packets from class i. Denote by α_i the ratio between the maximum length to the minimum length of packets in class i. Since for every i we have $\alpha_i \leq 2$, using linearity of expectation, we conclude that the above randomized non-preemptive algorithm is $O(\log n)$-competitive.

The Deterministic Case

The non-preemptive simple strategy applied above will not do in the deterministic setting. To see this, consider an input sequence consisting of all zero slack packets. One packet which needs to traverse the entire network, followed by a sequence of $(n-2)$ unit-path-length non-intersecting packets, each intersecting the path of the first packet on a different link. It follows that any non-preemptive deterministic algorithm can be $\Omega(n)$-competitive at best. We apply a different method for the deterministic case to balance between "long" and "short" packets. We analyze in Theorem 2.2.3 the competitive ratio guarantee of our algorithm, which we call MT (See Algorithm 2.1 below).

Algorithm 2.1 Algorithm MT

Given a new packet p just arrived,

1: **if** there exists a wave c eligible for p such that p doesn't intersect any currently scheduled packet on c **then**
2: schedule p on c
3: **else**
4: let c be the earliest eligible wave for p
5: **while** c is still eligible for p and p is not yet scheduled **do**
6: let q be the first (i.e., leftmost) packet scheduled on c which intersects p
7: **if** $|p| \leq |q|/2$ and $t_p \leq t_q$ **then**
8: replace q by p ▷ p evicts q
9: **end if**
10: $c \leftarrow c + 1$
11: **end while**
12: **end if**

We say that packet p *evicts* packet q if the condition in line 7 holds and q is replaced by p. Let us first make sure that the algorithm is well defined, and indeed produces a feasible schedule.

Lemma 2.2.2. *For any sequence of h packets, MT produces a feasible schedule.*

Proof. Proof by induction on h. For $h = 0$, the claim clearly holds. Assume the claim is true for any sequence of $h-1$ packets. Let p be the h'th packet introduced. If p is scheduled on a wave c such that p doesn't intersect any currently scheduled packet on c, then the schedule remains feasible. Otherwise, assume for every wave c eligible for p, there are scheduled packets intersecting p on c. If p is not scheduled by MT, then clearly the schedule remains feasible.

Otherwise, let c be the wave on which MT schedules p. Let S_p be the set of packets intersecting p on c, and let $q \in S_p$ be the first packet (i.e., leftmost packet) which intersects p on c. Since p is scheduled on c, by the condition in line 7 it follows that $t_p \leq t_q$, hence $S_p = \{q\}$, and therefore since q is evicted in favor of p, all the packets intersecting p on c are evicted, which results in a feasible schedule. □

We now turn to show the competitive ratio guarantee of MT.

Theorem 2.2.3. *Algorithm MT is an $O(\min\{\log\alpha, R\})$-competitive algorithm, where α is the ratio between the longest and shortest packets, and R is the number of different packet lengths.*

Proof. Let A be the set of packets scheduled by MT and O be the set of packets scheduled by some optimal schedule. Denote by N the set of packets never scheduled on any wave by MT. We distinguish between two types of packets in $O \setminus A$.

Packets scheduled. Consider a packet $p \in (O \setminus A) \cap \overline{N}$. Let c be the wave on which p was scheduled. Packet p was not successfully sent by A, and therefore was evicted from c by some packet q. Note that this can only happen if the condition of line 7 is met. Note that q is not necessarily in A either, since q might have been later evicted by a packet q'. However, notice that continuing this scenario eventually results in a packet that is successfully sent by A, since the line is of finite length, and in every time step we have a finite number of packets' arrivals. Denote any such maximal sequence by $q_1, \ldots, q_k$, where q_1 is a packet scheduled on c without evicting any other packet, and q_k is a packet eventually sent by A. We therefore have, by the condition of line 7, $|q_{i+1}| \leq |q_i|/2$ for all $i = 1, \ldots, k-1$.

Let us map each such p to its corresponding q_k. Each p that maps to a specific q_k, is mapped via one of the packets q_i that evicts it. Notice that the proof of Lemma 2.2.2 implies that any such q_i is responsible for evicting at most one packet from $(O \setminus A) \cap \overline{N}$. We therefore have a one-to-one correspondence between packets in $(O \setminus A) \cap \overline{N}$ and packets in any such maximal sequence. Let us turn to bound the size of each sequence:

$$m \leq |q_k| \leq 2^{-(k-1)} |q_1| \leq 2^{-(k-1)} M$$

which in turn yields

$$k \leq \log\alpha + 1. \tag{2.1}$$

It follows that $\left|(O \setminus A) \cap \overline{N}\right| \leq (k-1)|A|$ since for each sequence we have one packet that is eventually sent by A.

Packets never scheduled. Consider a packet $p \in (O \setminus A) \cap N$. Packet p was never scheduled because on each wave c eligible for p the condition of line 7 was not met. This specifically holds for the wave c on which O schedules p. Let q be the packet scheduled by A on c specified by the algorithm in line 6, preventing p from being scheduled on c. We show that any such q may prevent the schedule of at most 3 packets in $(O \setminus A) \cap N$. There might be at most one packet in $(O \setminus A) \cap N$ that is rejected by A due to its end point being later than that of the conflicting packet q scheduled by A. This is because O produces a valid schedule, so there is at most one packet using c on any edge, specifically at most one using the edge leaving the endpoint of q. Furthermore, notice that such a packet q can be responsible for the rejection of at most 2 packets in $(O \setminus A) \cap N$ that suffer from excess length. This is because all packets in $(O \setminus A) \cap N$ are

successfully scheduled on c in the optimal schedule, and therefore do not intersect on c. Since every such packet has length greater than $|q|/2$, there can be at most two such packets. It therefore follows that any such q may prevent the schedule of at most 3 packets in $(O \setminus A) \cap N$. The same maximal sequences identified in the analysis of the packets in $(O \setminus A) \cap \overline{N}$ occur here. There are at most k such packets in any such sequence, where each one is "responsible" for the non-scheduling of at most 3 packets. It follows that $|(O \setminus A) \cap N| \leq 3k\,|A|$.

We can now conclude the proof of the theorem. The above analysis yields

$$\begin{aligned} |O| &\leq |(O \setminus A) \cap N| + \left|(O \setminus A) \cap \overline{N}\right| + |A| \\ &\leq (k - 1 + 3k + 1)\,|A| \\ &= 4k\,|A|\,. \end{aligned}$$

Note that by the condition of line 7, any maximal sequence $q_1, \ldots, q_k$ satisfies $|q_1| > |q_2| > \ldots > |q_k|$, hence $k \leq R$. Combining this with Eq. (2.1) gives a competitive ratio of $O(\min\{\log \alpha, R\})$, which completes the proof. □

MT has running time of $O(\delta n)$ per packet, where δ is the maximal slack of any packet in the sequence, and n is the network size. Note that MT need not know the values of α or R in advance.

2.2.4 A Tight Example for MT

We now give an example showing that the above analysis is tight, up to a constant factor. I.e., the above algorithm cannot achieve a performance superior to $\Omega(\log n)$. Assume that $n = 2^k$ and let $r = k/2 - 1$. Define $x_i = \frac{1}{2^i}$. We therefore have $x_{i+1} = x_i/2$.

We consider two series of packets: $P = \{p_1, p_2, \ldots, p_r\}$ and $P' = \{p'_1, p'_2, \ldots, p'_r\}$, all with zero slack, where each packet is defined by its release time and its path:

- Packet p_i: release time $s_i = n(1 - x_i)$ and path $[s_i, n]$.
- Packet p'_i: release time $s'_i = n(1 - x_i) + 1 + i$ and path $[s'_i, s'_i + nx_{i+1} + 1]$.

Figure 2.2 shows an outline of the above sequence.

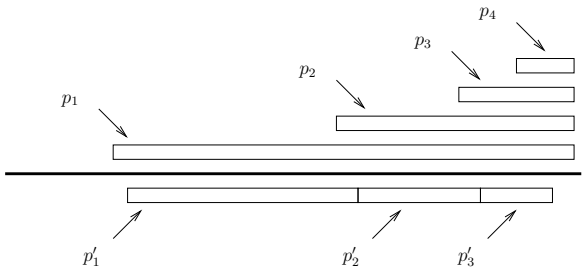


Figure 2.2: Outline of the sequence showing MT is $\Omega(\log n)$-competitive. The order induced by the release times is $(p_1, p'_1, p_2, p'_2, \ldots, p_r, p'_r)$.

Observation. For all i, $s_i < s'_i < s_{i+1}$. The first inequality follows from the definition, whereas the second follows from the fact that

$$\begin{aligned} s_{i+1} - s'_i &= -nx_{i+1} + nx_i - 1 - i \\ &= nx_{i+1} - 1 - i \\ &= 2^{k-(i+1)} - (i+1) > 0 \end{aligned}$$

for all $i \leq k/2 - 1 = r$.

Lemma 2.2.4. *For every i, if p_i is scheduled by MT at the end of time s_i, then p'_i is rejected by MT.*

Proof. Assume p_i is currently scheduled by MT. By the previous observation, the next packet in the sequence is p'_i. Since

$$|p_i| = nx_i = 2nx_{i+1} < 2(nx_{i+1} + 1) = 2\left|p'_i\right|,$$

then by the condition in line 7, p'_i is rejected by MT. □

Lemma 2.2.5. *For every i, if p_i is scheduled by MT at the end of time s_i, then upon the arrival of p_{i+1}, MT preempts p_i and schedules p_{i+1} instead.*

Proof. Assume p_i is scheduled by MT at the end of time s_i. By Lemma 2.2.4, p'_i, which is the next packet in the sequence, is rejected. The following packet is p_{i+1}, for which we have $|p_i| = x_i \geq 2x_{i+1} = 2|p_{i+1}|$, and in addition p_{i+1} doesn't terminate after p_i. By the condition in line 7, p_i is preempted by MT and p_{i+1} is scheduled in its place. □

Lemma 2.2.6. *MT finishes scheduling only one packet from P, while there exists a scheduling that schedules all the packets in P'.*

Proof. Since MT starts by scheduling p_1, then by Lemmas 2.2.4 and 2.2.5 it finishes scheduling only p_r. On the other hand notice that we can schedule all the packets in P'. Since the end point of p'_i is

$$\begin{aligned} s'_i + nx_{i+1} + 1 &= n(1 - x_i) + 1 + i + nx_{i+1} + 1 \\ &= n(1 - x_{i+1}) + 1 + (i+1) \\ &= s'_{i+1}, \end{aligned}$$

its path does not intersect with that of p'_{i+1}'s. □

Since $|P'| = \Omega(\log n)$, this example shows our analysis is tight up to a constant factor.

2.3 Non-Uniform Weights

2.3.1 Maximum Network Utilization

Assume that every packet p has weight $w_p = |p|$, and recall that our goal is to maximize the sum of the weights of delivered packets. This setting corresponds to optimizing network utilization.

Unlike the case of uniform weights, the idea here is to prefer longer packets, which gives a better utilization of the network. Let ϕ denote the golden ratio [2]. Consider the following algorithm for the problem, which we call MNU (see Algorithm 2.2 below).

Algorithm 2.2 Algorithm MNU

Given a new packet p just arrived,

if there exists a wave c eligible for p such that p doesn't intersect any currently scheduled packet on c **then**
    schedule p on c
else
    let c be the earliest eligible wave for p
    while c is still eligible for p and p is not yet scheduled **do**
        let S_p be the set of packets scheduled on c which intersects p.
        if $|p| \geq \phi \cdot \max_{q \in S_p} |q|$ **then**
            replace S_p by p ▷ p evicts S_p
        end if
        $c \leftarrow c + 1$
    end while
end if

MNU is an adaptation to our model of the algorithm given by Garay *et al.* in [32], for the problem of call admission, where a call's value is its route length.

We say packet p was *rejected by packet* q if q is the packet with maximal length in S_p, and p is rejected by the algorithm. In case more than one such packet exists, we choose one of them arbitrarily. We will sometimes abuse notation, referring to a packet as the set of its edges and to a set of edges as the set of intervals defined by them. Assume the packets arrived in the order $p_1, \ldots, p_k$. We first introduce some notation. For every $1 \leq i \leq k$, and every wave c, let $A^c(i)$ be the set of packets scheduled on c after the arrival of the i'th packet. For every packet $p \in A^c(i)$, let us denote the following:

- S_p^c - the set of packets preempted by MNU in order to schedule p (might be empty).
- T_p^c - the transitive closure of S_p^c, i.e., $T_p^c = \bigcup_{q \in S_p^c} T_q^c$. This set is defined immediately after p arrives and remains unchanged thereafter, since it only depends on packets scheduled on c which arrived prior to the arrival of p.
- $R_p^c(i)$ - the set of packets up to the i'th packet, rejected by packets in $T_p^c \cup \{p\}$.
- $I_p^c(i)$ - the collection of all edges in the paths of packets in $T_p^c \cup R_p^c(i) \cup \{p\}$.

Lemma 2.3.1. *For every wave c, every i, and every $p \in A^c(i)$,*

$$I_p^c(i) \subseteq [s_p - \phi\,|p|\,, t_p + \phi\,|p|].$$

[2] $\phi = \frac{1+\sqrt{5}}{2}$

Proof. Clearly, the scheduling and preemption of packets on any wave c is of no consequence to packets scheduled on waves other than c. We may therefore deal with each wave independently. Let c be any wave. We prove the claim by induction on i. The claim trivially holds for $i = 0$. Assume the claim holds for $i - 1$. Let p be the i'th packet that arrived. If c is not eligible for p then the claim clearly holds, so assume c is eligible for p.

Assume first that p is scheduled on c, and does not intersect any currently scheduled packet on c. In this case, for every packet $q \in A^c(i)$ other than p, $I^c_q(i) = I^c_q(i-1)$ and the induction hypothesis ensures the required result. For p we have $S^c_p = T^c_p = R^c_p(i) = \emptyset$, hence $I^c_p(i) = \{e | e \text{ is in } p\text{'s path}\}$, and the claim trivially holds.

Assume next that p is not scheduled on c. Let q be the packet responsible for rejecting p. Hence $q = \arg\max_{w \in S^c_p} |w|$. We need to show that $p \subseteq [s_q - \phi |q| , t_q + \phi |q|]$. Assume the contrary. Therefore p's path contains a point to the left of $s_q - \phi |q|$, or it contains a point to the right of $t_q + \phi |q|$. Since p was rejected because of q, clearly p and q intersect. Hence, $|p| > \phi |q|$, contradicting the fact that p was rejected because of q, and should therefore satisfy $|p| \leq \phi \cdot \max_{w \in S^c_p} |w| = \phi |q|$.

The last case to consider is the case where p is scheduled on c, and preempts the packets in S^c_p. We only need concern ourselves with p, as for every packet $q \in A^c(i)$ other than p, $I^c_q(i) = I^c_q(i-1)$. We will show that for every packet $q \in S^c_p$ preempted by p, $I^c_q(i-1) \subseteq [s_p - \phi |p| , t_p + \phi |p|]$, which will complete our proof. Since $q \in S^c_p$, p and q intersect. Furthermore, since q was preempted by p we have that $|q| \leq |p|$. One of the following must therefore be true: either $s_p \leq s_q < t_p$, or $s_p < t_q \leq t_p$. In both cases we have $[s_q - \phi |q| , t_q + \phi |q|] \subseteq [s_p - (|q| + \phi |q|), t_p + (|q| + \phi |q|)]$. Since $|p| \geq \phi |q|$, we have $|q| + \phi |q| \leq |p| / \phi + |p| = \phi |p|$, where the last equality follows from the defintion of ϕ. This concludes the proof of the lemma. □

We will use the above lemma, to analyze the performance of algorithm MNU.

Theorem 2.3.2. *MNU is a $(2\phi+1)$-competitive[3] algorithm for the problem of maximum network utilization, where ϕ denotes the golden ratio.*

Proof. An immediate consequence of Lemma 2.3.1 is the fact that for every wave c, every i, and every $p \in A^c(i)$, $\left|I^c_p(i)\right| \leq (1 + 2\phi) |p|$. Intuitively, this means that by scheduling p we have lost at most a factor of $(1 + 2\phi)$ in the objective function due to packets previously rejected or preempted to accommodate for the scheduling of p. Given a set of packets X, let $U(X) = \sum_{p \in X} |p|$. Consider the set of packets O scheduled in some optimal solution. Denote by A the set of packets that MNU schedules. Every packet in the sequence contributes its edges to at least one set $I^c_p(n)$, for some wave c, and some $p \in A^c(n)$ (since every packet is either scheduled, or was rejected or preempted). In particular every packet in O contributes its edges to at least one such set. We therefore have

$$\begin{aligned} U(O) &\leq \sum_{\text{wave}c} \sum_{p \in A^c(n)} \left|I^c_p(n)\right| \\ &\leq \sum_{\text{wave } c} \sum_{p \in A^c(n)} (1 + 2\phi) |p| \\ &= (1 + 2\phi) U(A) \end{aligned}$$

which completes the proof of the theorem. □

[3] $2\phi + 1 \sim 4.236$

Baruah *et al.* ([19]) present a lower bound of 4 for a problem of online task scheduling on a single machine, which applies to our model as well. It follows that any deterministic algorithm for our problem cannot have a competitive factor better than 4.

2.3.2 Arbitrary Weights

Clearly algorithm MNU appearing in Section 2.3.1 is guaranteed to produce a schedule which is within a factor of $(2\phi + 1)\beta = O(\beta)$ from an optimal schedule. A lower bound of $\Omega(\beta)$ for arbitrary weights follows from a lower bound for the problem of online task scheduling on a single machine, appearing in [19]. It follows that algorithm MNU has the best competitive ratio one could hope for, up to a constant factor.

2.4 The Ring Topology

Our results readily extend to a ring network topology. To see this, notice that our algorithms for a linear network compute a packing of the packets on the waves. We therefore need only present an appropriate notion of waves for a ring topology, which we call *ring-waves*. Given these waves, our algorithms can be adapted in a straightforward manner to the ring topology.

A ring is characterized by an underlying digraph $G = (V, E)$, where $V = \{0, \ldots, n-1\}$ and $E = \{(i, i+1 \mod n) | 1 \leq i \leq n-1\}$. In the linear topology, we have an unbounded number of waves, each of finite length defined by the size of the network. In a ring topology, however, we have a finite number ring-waves, defined by the size of the network, where each ring-wave is of infinite length. Every ring-wave is specified by sequence of pairs (t, j), where t represents a time step, and j represents a node in the network. Ring-wave i corresponds to the set $\{(t, j) | t - j + i = 0 \mod n\}$. See Figure 2.3 for an illustration of the ring waves for a ring of size 6.

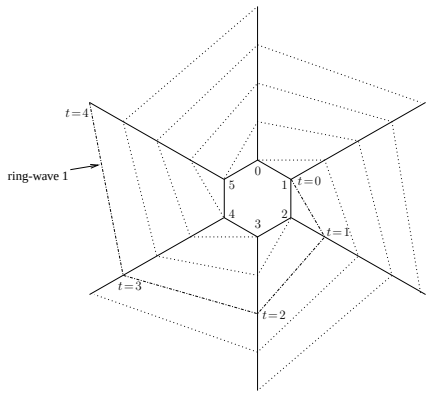


Figure 2.3: Geometric interpretation of ring-waves for a network of size 6. The hexagon represents the ring, and the solid lines represent time. Each wave is represented by a dotted line.

2.5 Discussion

We have presented the first online algorithms for the problem of bufferless time-constrained scheduling of packets in a linear network. These results extend to the ring topology as well. For the problem of maximum throughput, i.e., when packets have uniform weights, our algorithm achieves a competitive ratio of $O(\min\{\log\alpha, R\})$, where α is the ratio between the longest and shortest path lengths a packet has, and R is the number of different lengths of packet paths appearing in the input sequence. We additionally show that no online deterministic algorithm can achieve a competitive ratio better than 2 for this setting. We present a constant competitive algorithm for the problem of maximizing network utilization, where the weight of each packet is its length. For the case of arbitrary packet weights we give an algorithm with competitive ratio $O(\beta)$, where β is the ratio between the maximum and minimum weight-to-length ratios. Our algorithms for these cases are optimal up to a constant factor.

It would be interesting to try and close the gap between the upper and lower bounds for the problem of throughput maximization, as well as to see how rescheduling can affect the performance of such algorithms.

Chapter 3

Jitter Regulation for Multiple Streams in Capacitated Networks

3.1 Introduction

Contemporary network applications call for connections with stringent Quality-of-Service (QoS) demands. This gives rise to QoS networks that are able to provide guarantees on various parameters, such as the end-to-end delay, loss ratio, bandwidth, and jitter. The need for efficient mechanisms to provide smooth and continuous traffic is mostly motivated by the increasing popularity of interactive communication and in particular video/audio streaming.[1] The smoothness of such traffic is captured by the notion of *delay jitter* (or *Cell Delay Variation* [13]); namely, the difference between the maximal and minimal end-to-end delays of different fixed-size packets, henceforth referred to as *cells*.

Controlling traffic distortions within the network, and in particular jitter control, has the effect of moderating the traffic throughout the network [72]. This is important when a service provider in a QoS network must meet service level agreements (SLAs) with its customers. In such cases, moderating high congestion states in switches along the network results in the provider's ability to satisfy the guarantees to more customers [68].

Jitter control mechanisms have been extensively studied in recent years (see a survey in [72]). These are usually modelled as *jitter regulators* [39, 56, 73] that use internal buffers in order to shape the traffic, so that cells leave the regulator in the most periodic manner possible. Generally, such regulators calculate a hypothetical periodic schedule, and try to release cells accordingly. Upon arrival, cells are stored in the buffer until their planned release time, or until a buffer overflow occurs. This indicates a tradeoff between the buffer size and the best attainable jitter, i.e., as buffer space increases, one can expect to obtain a lower jitter.

In this chapter we investigate the problem of finding an optimal jitter release schedule, given a predetermined buffer size. This problem was first raised by Mansour and Patt-Shamir [56], who considered only a single-stream setting. However, in practice, jitter regulators handle mul-

[1] For example, 6.98 billion video streams were initiated by U.S. users during August 2006, while the U.S. streaming audience increased by 4 percent from July 2006 to reach 110.3 million streamers in August 2006, representing about 64 percent of the total U.S. Internet audience [25].

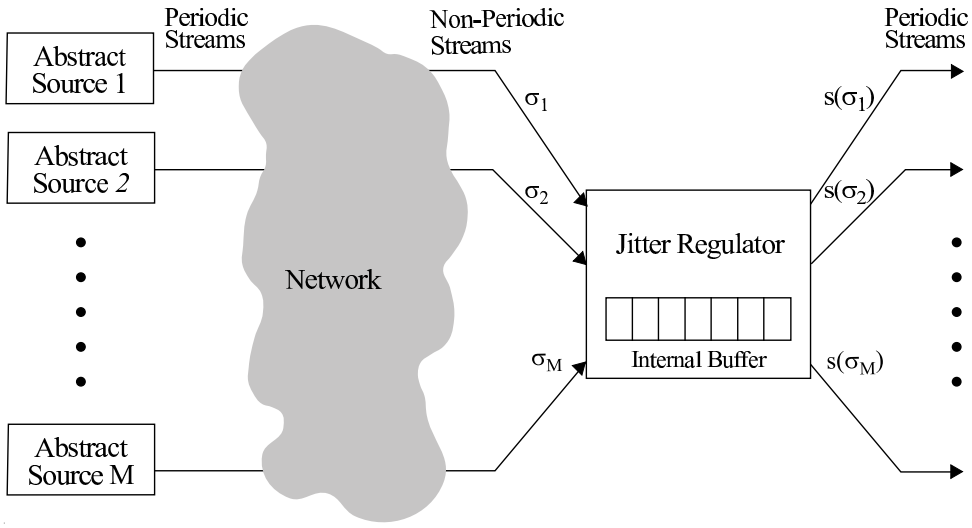


Figure 3.1: The multi-stream jitter regulation model

tiple streams simultaneously and must provide low jitter for each stream separately and independently. Furthermore, in real-life networks, links have finite capacities, which impose further limitations that should be taken into account by the regulator.

In the *multi-stream* model, the traffic arriving at the regulator is an interleaving of M streams originating from M independent abstract sources (see Figure 3.1). Each abstract source i sends a stream of fixed-size cells in a fully periodic manner, with inter-release time X^i, which arrive at a jitter regulator after traversing the network. Variable end-to-end delays caused by transient congestion throughout the network may result in such a stream arriving at the regulator in a non-periodic fashion. The regulator knows the value of X^i, and strives to release consecutive cells X^i time units apart, thus re-shaping the traffic into its original form, while respecting the capacity constraints of the outgoing link associated with stream i. Furthermore, the order in which cells are released by each abstract source is assumed to be respected throughout the network. This implies that the cells from the same stream arrive at the regulator in order (but not necessarily equally spaced), and the regulator should also maintain this order. We refer to this property as the *FIFO constraint*.

Note that the FIFO constraint should be respected in each stream independently, but not necessarily on all incoming traffic. This implies that in the multi-stream model, the order in which cells are released is not known *a priori*. This lack of knowledge is an inherent difference from the case where there is only one abstract source, and it poses a major difficulty in devising algorithms for multi-stream jitter regulation (as we describe in detail in Section 3.3).

3.1.1 Our Results

In this chapter we present algorithms and tight lower bounds for jitter regulation in this multiple streams environment, both in offline and online settings. This answers a primary question posed in [56].

We evaluate the performance of a regulator in the multi-stream model by considering the

maximum jitter obtained on any stream. We show that, somewhat surprisingly, the offline problem can be solved in polynomial time. This is done by characterizing a collection of optimal schedules, and showing that their properties can be used to devise an offline algorithm that efficiently finds a release schedule that attains the optimal jitter.

When considering the problem in the online setting, by sizing up the buffer to a size of $2MB$ and statically partitioning the buffer equally among the M streams, applying the algorithm described in [56, Algorithm B] on each stream separately yields an algorithm that obtains the optimal max-jitter possible with a buffer of size B. We show that such a resource augmentation cannot be avoided, by proving that any online algorithm needs a buffer of size at least MB in order to obtain a jitter within a bounded factor from the optimal jitter possible with a buffer of size B. We further show that these tight results (up to a factor of 2) also apply when the objective is to minimize the *average* jitter attained by the M streams. These results indicate that online jitter regulation does not scale well as the number of streams increases unless the buffer is sized up proportionally.

3.1.2 Previous Work

The problem of jitter control has received much attention in recent years, along with the increasing importance of providing QoS guarantees. A prime example is the *Differentiated Services (DiffServ)* architecture, in which there is a specific requirement to maintain low-jitter for *Expedited Forwarding (EF)* traffic [27].

Several algorithms have been proposed with the aim of providing traffic jitter control. Generally, all proposed algorithms are not work-conserving, i.e., they might delay releasing a cell even if there are cells in the buffer and the outgoing links are not fully utilized.

A jitter control algorithm which reconstructs the entire sequence at the destination using a predetermined maximum delay bound was proposed in [63]. The *Jitter-Earliest-Due-Date* algorithm proposed in [70] uses a predetermined maximum delay bound in order to calculate a deadline for every cell, such that it is released precisely upon its deadline. The *Stop-and-Go* algorithm proposed in [34] uses time frames of predetermined lengths in order to regulate traffic, such that cells arriving in the middle of a frame, are only made available for sending in the following time frame. The *Hierarchical-Round-Robin* algorithm proposed in [38] uses a framing strategy similar to the one used in the Stop-and-Go algorithm, but releases are governed by a round robin policy that sometimes allocates non-utilized release time-slots to other streams. For a more thorough survey of jitter control algorithms, see [72]. A slightly different line of research investigated jitter regulation in the Combined Input-Output Queue switch architecture, forcing the jitter regulator to obey additional constraints posed by the switching architecture [40].

The problem of jitter control in an adversarial setting has been discussed by Mansour and Patt-Shamir [56], where they consider a simplified *single-stream* model in which there is only a single abstract source. They present an efficient offline algorithm, which computes an optimal release schedule in these settings. They further devise an online algorithm, which uses a buffer of size $2B$, and produces a release schedule with the optimal jitter attainable with buffer of size B, and then show a matching lower bound on the amount of resource augmentation needed, proving that their online algorithm is optimal in this sense.

This model is later discussed by Koga [45] that deals with jitter regulation of a single stream with delay consideration. An optimal offline algorithm, and a nearly optimal online algorithm are presented for the case where a cell cannot be stored in the buffer for more than a predetermined amount of time.

3.1.3 Model

In this section we define the basic concepts of the multi-stream jitter regulation model, with bounded capacity links.

Definition 3.1.1. *Given a sequence of cells $\sigma = (p_i^\sigma)_{i=0}^n$ and a non-decreasing arrival function $a : \sigma \to \mathbb{R}^+$ such that cell p_i^σ arrives at time $a(p_i^\sigma)$, we define the following:*

1. *A* release schedule *for σ is a function $s : \sigma \to \mathbb{R}^+$ satisfying for every $p_i^\sigma \in \sigma$, $a(p_i^\sigma) \leq s(p_i^\sigma)$.*

2. *A release schedule s for σ is* B-feasible *if at any time t,*

$$|\{p_i^\sigma \in \sigma | a(p_i^\sigma) \leq t < s(p_i^\sigma)\}| \leq B.$$

   That is, there are never more than B cells in the buffer simultaneously.

3. *Given a link capacity $\lambda \in \mathbb{N}$, a release schedule s for σ is* capacity-feasible *if for any $p_j^\sigma \in \sigma$,*

$$s(p_{j+\lambda}^\sigma) \geq s(p_j^\sigma) + 1.$$

   That is, at any time t, at most λ cells are transmitted over the same link simultaneously.

4. *A release schedule is* feasible *if it is B-feasible and capacity-feasible.*

5. *The* delay jitter *of σ under a release schedule s is*

$$J^\sigma(s) = \max_{0 \leq i,k \leq n} \left\{ s(p_i^\sigma) - s(p_k^\sigma) - (i-k)X \right\}$$

   where X is the inter-release time of σ (i.e., X is the difference between the release times of any two consecutive cells from the abstract source).[2]

We first extend Definition 3.1.1 to an arrival sequence σ that is an interleaving of M streams $\sigma_1, \ldots, \sigma_M$. We denote by X^{σ_i} the inter-release time of stream σ_i, and by λ^{σ_i} the capacity of its outgoing link. An essential assumption is that $1/X^{\sigma_i} \leq \lambda^{\sigma_i}$, since otherwise the outgoing link cannot support the source's rate, and therefore cells must be dropped.

We assume for simplicity that all streams have the same inter-release time X, and the same outgoing link capacity λ; all our results extend immediately to the case where this does not hold.

Let p_j^σ denote the j'th cell (in order of arrival) of the interleaving of the streams σ, and let $p_j^{\sigma_i}$ denote the j'th cell of the single stream σ_i. A release schedule should obey a per-stream FIFO discipline, in which cells of the same stream are released in the order of their arrival.

Let $J^{\sigma_i}(s)$ be the jitter of a single stream σ_i obtained by a release schedule s. We use the following metric to evaluate multi-stream release schedules:

[2] Since the abstract source generates perfectly periodic traffic, this definition of delay jitter coincides with the notion of Cell Delay Variation.

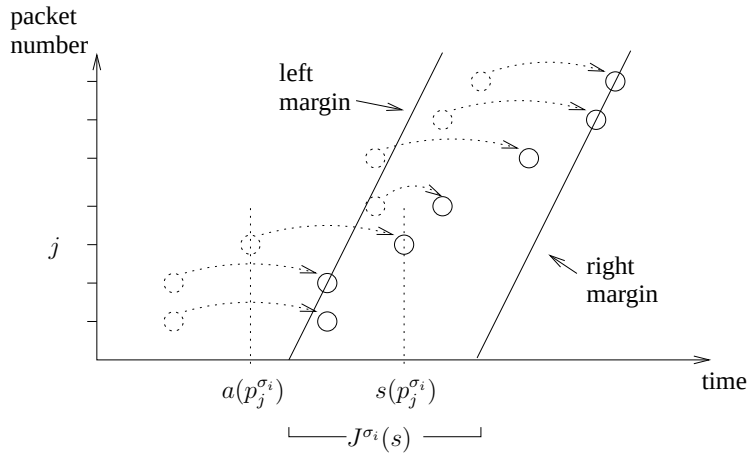


Figure 3.2: Outline of arrivals (dotted circles) and marked releases (full circles). The jitter of stream σ_i is the width of the band with slope $1/X$ enclosing all releases.

Definition 3.1.2. *The* max-jitter *of a multi-stream sequence $\sigma = \bigcup\{\sigma_1, \ldots, \sigma_M\}$ obtained by a release schedule s is the maximal jitter obtained by any of the streams composing the sequence; that is,*

$$\mathrm{MJ}^{\sigma}(s) = \max_{1 \le k \le M} J^{\sigma_k}(s).$$

In what follows, given any algorithm A, we denote by $J^{\sigma_i}(A)$ ($\mathrm{MJ}^{\sigma}(A)$) the jitter (max-jitter) corresponding to the schedule produced by A given stream σ_i (multi-stream σ). If an algorithm A fails to produce a feasible schedule for stream σ_i, we define $J^{\sigma_i}(A) = \infty$.

Geometric Intuition One can take a geometric view of delay jitter by considering a two dimensional plane where the x-axis denotes time and the y-axis denotes the cell number. We first consider the case of a single stream σ. Given a release schedule s, a point at coordinates (t, j) is marked if $s(p_j^{\sigma}) = t$ (see Figure 3.2). The *release band* is the band with slope $1/X$ that encloses all the marked points and has minimal *width*, where the width of the band is the maximal difference in the *x*-axis coordinates between its margins. The jitter obtained by s is the width of its release band, and therefore our objective is to find a schedule with the narrowest release band.

Under the multi-stream model, we associate every stream σ_i with a different color i. A point at coordinates (t, j) is colored with color i if $s(p_j^{\sigma_i}) = t$. Any schedule s induces a separate release band for each stream σ_i in σ that encloses all points with color i. Schedule s is therefore characterized by M release bands.

3.2 Online Multi-Stream Max-Jitter Regulation

As mentioned previously, there exists an online algorithm with buffer size $2MB$, which obtains the optimal max-jitter possible with a buffer of size B. In this section we show that this result is tight up to a factor of 2, by showing that in order to obtain a max-jitter within a bounded factor

from the optimal max-jitter possible with a buffer of size B, any online algorithm needs a buffer of size at least MB cells. Hence, in order to maintain any reasonable jitter performance, it is necessary to increase the buffer size in a linear proportion to the number of streams.

Theorem 3.2.1. *For every online algorithm* ALG *with an internal buffer of size smaller than* MB*, and for any* $T > 0$*, there exists an arrival sequence consisting of* M *streams, forcing* ALG *to have max-jitter at least* T*, while the optimal jitter possible with a buffer of size* B *is zero.*

Proof. Let ALG be an online algorithm with a buffer of size at most $MB - 1$. Consider the following arrival sequence σ: For every $0 \leq k \leq B - 1$, M cells arrive at the regulator at time $k \cdot X$, one for every stream.

Since the buffer size is at most $MB - 1$ and MB cells arrived by time $t' = (B - 1)X$, it follows that ALG releases a cell by time t', say of stream σ_i. Consider the following continuation for σ: Given some $T > 0$, in time $T' \geq t' + BX + T$, a single cell of stream σ_i arrives at the regulator.

Note that ALG releases the first cell of stream σ_i by time t', the last cell of stream σ_i cannot be sent prior to time T', and σ_i consists of $B + 1$ cells. Let s be the schedule produced by ALG. It follows that

$$\begin{aligned} J^{\sigma_i}(\mathrm{ALG}) &\geq s(p_B^{\sigma_i}) - s(p_0^{\sigma_i}) - (B - 0)X \\ &\geq T' - t' - BX \\ &\geq T + t' + BX - t' - BX = T, \end{aligned}$$

which can be arbitrarily large. It follows also that $\mathrm{MJ}^{\sigma}(\mathrm{ALG}) \geq T$. On the other hand, note that for any choice of T, the optimal max-jitter possible with a buffer of size B is zero: Every cell of a stream other than σ_i is released immediately upon its arrival, and for every $0 \leq j \leq B$, cell $p_j^{\sigma_i}$ is released in time $T - (B - j)X$. Since every stream other than σ_i does not consume any buffer space, it is easy to verify that at every time t, there are at most B cells in the buffer. In addition, each stream in σ obeys its capacity constraint since $1/X \leq \lambda$. Clearly, every stream attains zero jitter by this release schedule. □

Theorem 3.2.1 implies that in case the buffer size is smaller than MB, there are scenarios in which an optimal schedule attains zero jitter for all streams, while any online algorithm can be forced to produce a schedule with arbitrarily large max-jitter. This fact immediately implies that even if the objective is to minimize the average jitter obtained by the different streams, the same lower bound holds. Since the online algorithm, which statically partitions a buffer, minimizes the jitter of each stream independently, it clearly minimizes the overall average jitter as well, thus providing a matching upper bound (up to a factor of 2).

Moreover, we are able to prove a more general lower bound:

Theorem 3.2.2. *For every online algorithm* ALG *with an internal buffer size smaller than*

$$\max\{MB, M(B-1) + B + 1\},$$

there exists an arrival sequence consisting of M *streams, such that* ALG *attains max-jitter strictly greater than the optimal jitter possible with a buffer of size* B*.*

Proof. Let ALG be an online algorithm with a buffer of size at most $M(B-1)+B$. Consider the following arrival sequence σ: For every $0 \leq k \leq B-1$, M cells arrive at the regulator at time $k \cdot X$, one for every stream. The sequence stops if ALG releases a cell before time $t' = BX$.

If ALG releases a cell before time t', the claim follows from the proof of Theorem 3.2.1.

Therefore, assume now that ALG does not release any cells before time t', implying that in time t' there are MB cells in the buffer. Note that this implies that ALG has buffer size at least MB. Consider the following continuation for σ: In time t', $B+1$ cells of stream σ_1 arrive at the regulator.

Since ALG has a buffer of size at most $M(B-1)+B = MB+B-M$, and before time t' ALG has not released any of the $MB+B+1$ cells in σ, it must release at least $M+1$ cells in time t'. By the pigeonhole principle it follows that at least two of the released cells correspond to the same stream, say σ_i. If $\lambda = 1$, ALG cannot produce a feasible schedule, and therefore $\mathrm{MJ}^{\sigma}(\mathrm{ALG}) = \infty$. If $\lambda \geq 2$, the release schedule produced by ALG, denoted by s, can be feasible, but the jitter attained by stream σ_i is at least

$$s(p_0^{\sigma_i}) - s(p_1^{\sigma_i}) - (0-1)X = t' - t' - (0-1)X = X,$$

and therefore $\mathrm{MJ}^{\sigma}(\mathrm{ALG})$ is strictly greater than zero. On the other hand, the optimal max-jitter possible with a buffer of size B is zero. To see this, consider the following release schedule: Every cell of a stream other than σ_1 is released immediately upon its arrival, and for every $0 \leq j \leq 2B$, cell $p_j^{\sigma_1}$ is released in time $t' - (B-j)X$. Similarly to the previous case, every stream obtains a zero jitter by this release schedule, and no more than B cells are stored simultaneously in the buffer since every cell, except the last B cells of stream σ_1, is sent immediately upon arrival, and no two cells of the same stream are sent simultaneously. □

Note that Theorem 3.2.2 implies that a greater resource augmentation is required, compared to Theorem 3.2.1, while Theorem 3.2.1 implies unbounded competitiveness whereas Theorem 3.2.2 only implies no online algorithm can obtain the *optimal* jitter. Furthermore, the lower bound described in Theorem 3.2.2 exactly coincides with the result of the single stream model (i.e., $M = 1$) with no capacity constraint on the outgoing link [56].

3.3 An Efficient Offline Algorithm

This section presents an efficient offline algorithm that generates a release schedule with optimal max-jitter.

Note that when imposing a capacity constraint on the outgoing link, some arrival sequences may prohibit any schedule from being feasible; For example, if a burst of $B+\lambda+1$ cells arrive at the regulator simultaneously, the link capacity constraint and the B-feasibility constraint lay in contradiction. The following lemma characterizes arrival sequences with feasible release schedules:

Lemma 3.3.1. *There exists a feasible schedule for a sequence $\sigma = \bigcup\{\sigma_1, \ldots, \sigma_M\}$ if and only if the greedy schedule s defined by*

$$s_{greedy}(p_j^{\sigma_i}) = \begin{cases} a(p_j^{\sigma_i}) & i = 1, \ldots, M \quad j = 0, \ldots, \lambda - 1 \\ \max\left\{a(p_j^{\sigma_i}), s_{greedy}(p_{j-\lambda}^{\sigma_i}) + 1\right\} & i = 1, \ldots, M \quad j = \lambda, \ldots \end{cases}$$

is B-feasible.

Proof. Assume s_{greedy} is B-feasible. Since s_{greedy} clearly satisfies the capacity constraint and the FIFO constraint, s_{greedy} is a feasible schedule.

On the other hand, assume there is a feasible schedule s' for the capacitated problem. For every stream σ_i, we prove by induction on the cell number j that for every $p_j^{\sigma_i}$, $s_{\text{greedy}}(p_j^{\sigma_i}) \leq s'(p_j^{\sigma_i})$. For $j = 0, \ldots, \lambda - 1$ the claim follows from the arrival feasibility of s'. Assume the claim holds for $m < j$. If $s_{\text{greedy}}(p_j^{\sigma_i}) = a(p_j^{\sigma_i})$ then the claim follows from the arrival feasibility of s'. Otherwise, $s_{\text{greedy}}(p_j^{\sigma_i}) = s_{\text{greedy}}(p_{j-\lambda}^{\sigma_i}) + 1$. By our induction hypothesis, $s_{\text{greedy}}(p_{j-\lambda}^{\sigma_i}) \leq s'(p_{j-\lambda}^{\sigma_i})$. Since s' satisfies the capacity constraint we have that $s'(p_j^{\sigma_i}) \geq s'(p_{j-\lambda}^{\sigma_i}) + 1$. Combining these inequalities, the claim follows. The claim shows that for any time t, any set of cells residing in the buffer in time t according to s_{greedy}, also resides in the buffer in time t according to s'. Since s' is B-feasible, it follows that s_{greedy} is B-feasible as well. □

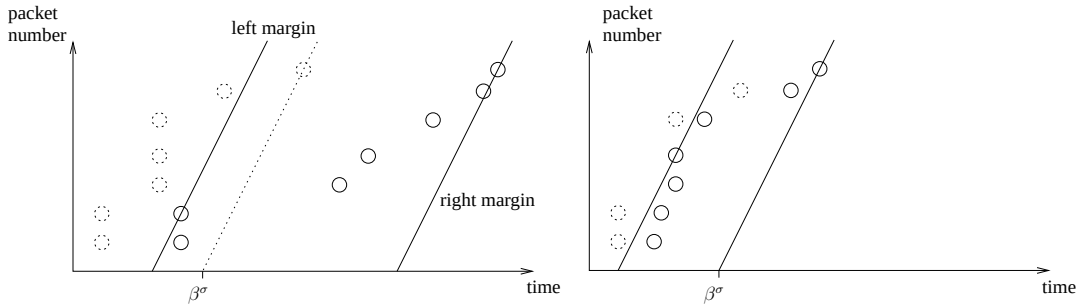
Given a sequence σ that is an interleaving of M streams and has a feasible release schedule, consider a total order $\pi = (p'_0, \ldots, p'_n)$ on the release schedule of cells in σ that respects the FIFO order in each stream separately. The release schedule, which attains the optimal max-jitter and respects π, can be found using similar arguments to the ones in [56, Algorithm A]: Essentially, cell p'_j can be stored in the buffer only until cell p'_{j+B} arrives, imposing strict bounds on the release time of each cell[3]. In particular, it follows that for every sequence σ that has a feasible release schedule, there exists an optimal release schedule. Unfortunately, it is computationally intractable to enumerate over all possible total orders, hence a more sophisticated approach should be considered.

We first discuss properties of schedules that achieve optimal max-jitter. We then show that these properties allow to find an optimal schedule (or decide that no feasible schedule exists) in polynomial time, based solely on the cells' arrival times, and the parameters X and B.

For every cell $p_j^{\sigma_i}$ of any stream σ_i, one can intuitively consider $t = a(p_j^{\sigma_i}) - jX$ as the time at which $p_0^{\sigma_i}$ should be sent, so that $p_j^{\sigma_i}$ is sent immediately upon its arrival, in a perfectly periodic release schedule. For any stream σ_i, denote by $\beta^{\sigma_i} = \max_j \left\{a(p_j^{\sigma_i}) - jX\right\}$. From a geometric point of view, β^{σ_i} is a lower bound on the intersection between the time axis and the right margin of any release band (see Figure 3.3(a)), since otherwise the cell defining β^{σ_i} would have been released prior to its arrival.

Given a release schedule s for a sequence σ, a stream $\sigma_i \subseteq \sigma$ is said to be *aligned* in s if there is no cell $p_k^{\sigma_i} \in \sigma_i$ such that $s(p_k^{\sigma_i}) > \beta^{\sigma_i} + kX$. Clearly, if σ_i is aligned in s, then any cell $p_j^{\sigma_i}$ that defines β^{σ_i} satisfies $s(p_j^{\sigma_i}) = a(p_j^{\sigma_i})$. Geometrically, the right margin of a release band corresponding to an aligned stream σ_i intersects the time axis in point $(\beta^{\sigma_i}, 0)$ (see Figure 3.3(b)).

[3]The algorithm in [56, Algorithm A] does not deal with capacity-constraint, but these can be easily incorporated.



(a) non-aligned schedule (b) aligned schedule

Figure 3.3: Outline of arrivals (dotted circles) and marked releases (full circles).

A release schedule s for σ is said to be *aligned*, if every stream is aligned in s. The following lemma shows that one can iteratively align the streams of an optimal schedule without increasing the overall jitter:

Lemma 3.3.2. *For every sequence σ, there exists an optimal aligned schedule s.*

Proof. Given an optimal schedule s' for sequence σ with $\ell < M$ aligned streams, we prove that s' can be changed into an aligned schedule (i.e. with M aligned streams), maintaining its optimality.

We first show that s' can be altered into an optimal schedule with $\ell + 1$ aligned streams. Let σ_i be one of the non-aligned streams in s', and consider the following schedule $\overline{s}$:

$$\overline{s}(p_j^{\sigma_k}) = \begin{cases} \min\left\{s'(p_j^{\sigma_k}), \beta^{\sigma_k} + jX\right\} & k = i \\ s'(p_j^{\sigma_k}) & k \neq i \end{cases}$$

Clearly, for every stream other than σ_i the schedule remains unchanged; therefore, it suffices to consider only stream σ_i. Since $s'(p_j^{\sigma_i}) \geq a(p_j^{\sigma_i})$ and $\beta^{\sigma_i} + jX \geq a(p_j^{\sigma_i})$, $\overline{s}$ is a release schedule and it can easily be verified that $\overline{s}$ satisfies the FIFO constraint. Schedule $\overline{s}$ is B-feasible, since s' is B-feasible and for any cell $p_j^{\sigma_i}$, $\overline{s}(p_j^{\sigma_i}) \leq s'(p_j^{\sigma_i})$.

In order to prove that schedule $\overline{s}$ is capacity-feasible, we consider any two cells $p_j^{\sigma_i}, p_{j+\lambda}^{\sigma_i}$ and show that $\overline{s}(p_{j+\lambda}^{\sigma_i}) \geq \overline{s}(p_j^{\sigma_i}) + 1$. We distinguish between four cases: If $\overline{s}(p_{j+\lambda}^{\sigma_i}) = s'(p_{j+\lambda}^{\sigma_i})$ and $\overline{s}(p_j^{\sigma_i}) = s'(p_j^{\sigma_i})$ the claim immediately follows from feasibility of s'. In case $\overline{s}(p_{j+\lambda}^{\sigma_i}) = \beta^{\sigma_k} + (j+\lambda)X$ and $\overline{s}(p_j^{\sigma_i}) = \beta^{\sigma_k} + jX$ then $\overline{s}(p_{j+\lambda}^{\sigma_i}) - \overline{s}(p_j^{\sigma_i}) = \lambda X \geq 1$, since $\lambda \geq 1/X$. If $\overline{s}(p_{j+\lambda}^{\sigma_i}) = s'(p_{j+\lambda}^{\sigma_i})$ and $\overline{s}(p_j^{\sigma_i}) = \beta^{\sigma_k} + jX$ then $\overline{s}(p_{j+\lambda}^{\sigma_i}) - \overline{s}(p_j^{\sigma_i}) \geq s'(p_{j+\lambda}^{\sigma_i}) - s'(p_j^{\sigma_i}) \geq 1$, by the definition of $\overline{s}$. Finally, if $\overline{s}(p_{j+\lambda}^{\sigma_i}) = \beta^{\sigma_k} + (j+\lambda)X$ and $\overline{s}(p_j^{\sigma_i}) = s'(p_j^{\sigma_i})$ then $\overline{s}(p_{j+\lambda}^{\sigma_i}) - \overline{s}(p_j^{\sigma_i}) \geq \beta^{\sigma_k} + (j+\lambda)X - \beta^{\sigma_k} + jX = \lambda X \geq 1$ by the definition of $\overline{s}$.

Stream σ_i is aligned in $\overline{s}$, since every cell $p_j^{\sigma_i}$ satisfies $\overline{s}(p_j^{\sigma_i}) \leq \beta^{\sigma_i} + jX$. Hence, $\overline{s}$ has $\ell + 1$ aligned stream.

In order to prove that $\overline{s}$ is optimal, it suffices to show that $\overline{s}(p_j^{\sigma_i}) - \overline{s}(p_m^{\sigma_i}) - (j-m)X \leq J^{\sigma_i}(s')$ for every two cells $p_j^{\sigma_i}, p_m^{\sigma_i} \in \sigma_i$. Assume without loss of generality that $\overline{s}(p_j^{\sigma_i}) - jX \geq \overline{s}(p_m^{\sigma_i}) - mX$. If this does not hold then our term is negative, and we can simply switch the roles of $p_j^{\sigma_i}$ and $p_m^{\sigma_i}$. We distinguish between four possible cases: In the first case where

$\overline{s}(p_j^{\sigma_i}) = s'(p_j^{\sigma_i})$ and $\overline{s}(p_m^{\sigma_i}) = s'(p_m^{\sigma_i})$ the result follows immediately for the definition of $J^{\sigma_i}(s')$. In the case where $\overline{s}(p_j^{\sigma_i}) = \beta^{\sigma_i} + jX$ and $\overline{s}(p_m^{\sigma_i}) = \beta^{\sigma_i} + mX$ the term is zero, which is not more than $J^{\sigma_i}(s')$ by definition. The third case to consider is when $\overline{s}(p_j^{\sigma_i}) = s'(p_j^{\sigma_i})$ and $\overline{s}(p_m^{\sigma_i}) = \beta^{\sigma_i} + mX$. This implies that $s'(p_j^{\sigma_i}) < \beta^{\sigma_i} + jX$, thus $s'(p_j^{\sigma_i}) - jX < \beta^{\sigma_i}$. Therefore $\overline{s}(p_j^{\sigma_i}) - jX = s'(p_j^{\sigma_i}) - jX < \beta^{\sigma_i} = \overline{s}(p_m^{\sigma_i}) - mX$, contradicting the assumption on $p_j^{\sigma_i}$ and $p_m^{\sigma_i}$. The last case to consider is when $\overline{s}(p_j^{\sigma_i}) = \beta^{\sigma_i} + jX$ and $\overline{s}(p_m^{\sigma_i}) = s'(p_m^{\sigma_i})$: Similarly to the previous case, this implies that $\beta^{\sigma_i} < s'(p_j^{\sigma_i}) - jX$, and therefore $\overline{s}(p_j^{\sigma_i}) - \overline{s}(p_m^{\sigma_i}) - (j-m)X = \beta^{\sigma_i} - (s'(p_m^{\sigma_i}) - mX) < s'(p_j^{\sigma_i}) - jX - (s'(p_m^{\sigma_i}) - mX) \leq J^{\sigma_i}(s')$, as required.

Applying the same arguments repeatedly alters schedule s' into an aligned schedule and preserves its optimality. □

Next we show that the optimality of a schedule s is maintained even if cells that are stored in the buffer are released earlier, as long as their new release time satisfies FIFO order and remains within a release band of width $\mathrm{MJ}^{\sigma}(s)$:

Lemma 3.3.3. *Let s be an optimal schedule for sequence σ. Then, for every stream $\sigma_i \subseteq \sigma$ and for every $J \in [J^{\sigma_i}(s), \mathrm{MJ}^{\sigma}(s)]$, the new schedule*

$$s'(p_j^{\sigma_k}) = \begin{cases} \max\left\{a(p_j^{\sigma_k}), \beta^{\sigma_k} - J + jX\right\} & k = i, j < \lambda \\ \max\left\{a(p_j^{\sigma_k}), \beta^{\sigma_k} - J + jX, s'(p_{j-\lambda}^{\sigma_k}) + 1\right\} & k = i, j \geq \lambda \\ s(p_j^{\sigma_k}) & k \neq i \end{cases}$$

is B-feasible and $\mathrm{MJ}^{\sigma}(s') = \mathrm{MJ}^{\sigma}(s)$. Furthermore, if s is aligned then so is s'.

Proof. Since s' only changes the release schedule of stream σ_i, it clearly preserves the FIFO order, capacity-constraint and jitter of each stream other than σ_i. Furthermore, by its definition, schedule s' clearly satisfies the capacity constraint for σ_i.

We first show that s' respects the FIFO order of cells in σ_i. Assume by contradiction that $p_j^{\sigma_i}$ is the first cell in σ_i such that $s'(p_j^{\sigma_i}) > s'(p_{j+1}^{\sigma_i})$. If $s'(p_j^{\sigma_i}) = a(p_j^{\sigma_i})$ then its release time cannot be later than $a(p_{j+1}^{\sigma_i}) \leq s'(p_{j+1}^{\sigma_i})$. If $s'(p_j^{\sigma_i}) = \beta^{\sigma_i} - J + jX$ then $s'(p_j^{\sigma_i}) \leq \beta^{\sigma_i} - J + (j+1)X \leq s'(p_{j+1}^{\sigma_i})$. It follows that $s'(p_j^{\sigma_i}) = s'(p_{j-\lambda}^{\sigma_i}) + 1$. However, $s'(p_{j+1}^{\sigma_i}) \geq s'(p_{j+1-\lambda}^{\sigma_i}) + 1 \geq s'(p_{j-\lambda}^{\sigma_i}) + 1 = s'(p_j^{\sigma_i})$, where the first inequality follows from the capacity constraint and the second inequality follows from the assumption that $p_j^{\sigma_i}$ is the first cell to violate the FIFO order.

In order to bound the max-jitter of s', it suffices to show that $J^{\sigma_i}(s') \leq \mathrm{MJ}^{\sigma}(s)$. We first show by induction on j that $s'(p_j^{\sigma_i}) \leq \beta^{\sigma_i} + jX$. For the base case $j = 0$, both $\beta^{\sigma_i} - J$ and $a(p_j^{\sigma_i})$ are at most β^{σ_i} by the definition of β^{σ_i}. For $j > 0$, $a(p_j^{\sigma_i}) \leq \beta^{\sigma_i} + jX$ and $s'(p_{j-\lambda}^{\sigma_i}) + 1 \leq \beta^{\sigma_i} + (j-\lambda)X + 1 \leq \beta^{\sigma_i} + jX$ by the induction hypothesis and the fact that $\lambda > 1/X$.

Consider any pair of cells $p_a^{\sigma_i}, p_b^{\sigma_i} \in \sigma_i$. By the definition of s', $s'(p_a^{\sigma_i}) \geq \beta^{\sigma_i} - J + aX$. As we proved, $s'(p_b^{\sigma_i}) \leq \beta^{\sigma_i} + bX$. Hence, $s'(p_b^{\sigma_i}) - s'(p_a^{\sigma_i}) \leq J + (b-a)X$, which implies that $J^{\sigma_i}(s') = \max_{a,b}\left\{s'(p_b^{\sigma_i}) - s'(p_a^{\sigma_i}) - (b-a)X\right\} \leq J \leq \mathrm{MJ}^{\sigma}(s)$.

In order to show that s' is B-feasible, we first show that for every cell $p_j^{\sigma_i} \in \sigma_i$, $s'(p_j^{\sigma_i}) \leq s(p_j^{\sigma_i})$. The proof is by induction on the cell index j. The claim clearly holds for $j = 0$, since $J \geq J^{\sigma_i}(s)$. For $j > 0$, we consider the following three cases. If $s'(p_j^{\sigma_i}) = a(p_j^{\sigma_i})$, then the

claim follows from the arrival feasibility of s. If $s'(p_j^{\sigma_i}) = s'(p_{j-\lambda}^{\sigma_i}) + 1$, then by the induction hypothesis, $s'(p_j^{\sigma_i}) \leq s(p_{j-\lambda}^{\sigma_i}) + 1$, which is at most $s(p_j^{\sigma_i})$, by the capacity-feasibility of s. The last case to consider is when $s'(p_j^{\sigma_i}) = \beta^{\sigma_i} - J + jX$. In this case

$$\begin{array}{rcll} s'(p_j^{\sigma_i}) & = & \beta^{\sigma_i} - J + jX & \text{by the definition of } s' \\ & \leq & \beta^{\sigma_i} - J^{\sigma_i}(s) + jX & \text{since } J \in [J^{\sigma_i}(s), \mathrm{MJ}^{\sigma}(s)] \\ & = & a(p_k^{\sigma_i}) - kX - J^{\sigma_i}(s) + jX & \text{for } p_k^{\sigma_i} \text{ defining } \beta^{\sigma_i} \\ & \leq & s(p_k^{\sigma_i}) - (k-j)X - J^{\sigma_i}(s) & \text{since } a(p_k^{\sigma_i}) \leq s(p_k^{\sigma_i}) \\ & \leq & s(p_k^{\sigma_i}) - (k-j)X - & \\ & & \Big(s(p_k^{\sigma_i}) - s(p_j^{\sigma_i}) - (k-j)X\Big) & \text{by definition of } J^{\sigma_i}(s) \\ & \leq & s(p_j^{\sigma_i}) & \end{array}$$

We now turn to show that s' is B-feasible. Assume the contrary, and let t be any time in which a set P of more than B cells are stored in the buffer. Since the release schedule of any stream σ_k other than σ_i is identical under both s and s', every cell $p_j^{\sigma_k} \in P$, for $k \neq i$, is also stored in the buffer at time t under schedule s. Since for every cell $p_j^{\sigma_i} \in \sigma_i$, $s'(p_j^{\sigma_i}) \leq s(p_j^{\sigma_i})$, all cells $p_j^{\sigma_i} \in P$ are stored in the buffer at time t under schedule s as well. This contradicts the assumption that s is B-feasible.

We conclude the proof by showing that if s is aligned then s' is also aligned. Assume s is aligned. For any stream $\sigma_k \neq \sigma_i$ schedules s and s' are identical on σ_k, and therefore σ_k is aligned in s'. As shown above, for every cell $p_j^{\sigma_i} \in \sigma_i$, $s'(p_j^{\sigma_i}) \leq s(p_j^{\sigma_i})$. Since s is aligned, we are guaranteed that for every cell $p_j^{\sigma_i} \in \sigma_i$, $s(p_j^{\sigma_i}) \leq \beta^{\sigma_i} + jX$. Combining these two inequalities we obtain that σ_i is also aligned in s'. □

By iteratively applying Lemma 3.3.3 with $J = \mathrm{MJ}^{\sigma}(s)$ on all streams, we get:

Corollary 3.3.4. *Given an optimal aligned schedule s for sequence σ, the schedule defined by*

$$s'(p_j^{\sigma_k}) = \begin{cases} \max\left\{a(p_j^{\sigma_k}), \beta^{\sigma_k} - \mathrm{MJ}^{\sigma}(s) + jX\right\} & j < \lambda \\ \max\left\{a(p_j^{\sigma_k}), \beta^{\sigma_k} - \mathrm{MJ}^{\sigma}(s) + jX, s'(p_{j-\lambda}^{\sigma_k}) + 1\right\} & j \geq \lambda \end{cases}$$

is an optimal aligned schedule.

The following lemma bounds from below the release time of cells in a schedule. Intuitively, this lemma defines the left margin of the release band.

Lemma 3.3.5. *For any schedule s for sequence σ, every stream $\sigma_i \subseteq \sigma$, and every cell $p_j^{\sigma_i}$, $s(p_j^{\sigma_i}) \geq \beta^{\sigma_i} - J^{\sigma_i}(s) + jX$.*

Proof. Assume by contradiction that there exists a stream σ_i and a cell $p_j^{\sigma_i}$ such that $s(p_j^{\sigma_i}) < \beta^{\sigma_i} - J^{\sigma_i}(s) + jX$. Let $p_k^{\sigma_i}$ be the cell defining β^{σ_i}. Since $s(p_k^{\sigma_i}) \geq a(p_k^{\sigma_i})$,

$$\begin{array}{rcl} J^{\sigma_i}(s) & \geq & s(p_k^{\sigma_i}) - s(p_j^{\sigma_i}) - (k-j)X \\ & > & a(p_k^{\sigma_i}) - (\beta^{\sigma_i} - J^{\sigma_i}(s) + jX) - (k-j)X \\ & = & a(p_k^{\sigma_i}) - (a(p_k^{\sigma_i}) - kX) + J^{\sigma_i}(s) - jX - kX + jX = J^{\sigma_i}(s), \end{array}$$

which is a contradiction. □

Lemma 3.3.5 indicates an important property of aligned optimal schedules. In such schedules, the jitter of any stream can be characterized by the release time of a single cell, as depicted in the following corollary:

Corollary 3.3.6. *For any aligned schedule s for sequence σ and every stream $\sigma_i \subseteq \sigma$,*

$$J^{\sigma_i}(s) = \max_j \left\{ \beta^{\sigma_i} - s(p_j^{\sigma_i}) + jX \right\}.$$

The following lemma shows that at least one of the widest release bands, corresponding to some stream σ_i attaining the max-jitter, has its left margin determined by the following event: An arrival of a cell causing a buffer overflow (possibly of another stream), which necessitates some cell of σ_i to be released earlier than desired.

Lemma 3.3.7. *Let s be an aligned optimal schedule for sequence σ. There exists a stream $\sigma_i \subseteq \sigma$ that attains the max-jitter, and a cell $p_j^{\sigma_i}$ such that $s(p_j^{\sigma_i}) = \beta^{\sigma_i} - \mathrm{MJ}^{\sigma}(s) + jX$ and $s(p_j^{\sigma_i}) = a(p_\ell^{\sigma})$ for some cell $p_\ell^{\sigma} \in \sigma$.*

Proof. We show by contradiction that if the claim does not hold for an optimal aligned schedule, then such a schedule can be altered into a new schedule with max-jitter strictly less than the original schedule. Formally, consider an aligned optimal schedule s for σ. Let $\Sigma = \{\sigma_i \mid J^{\sigma_i}(s) = \mathrm{MJ}^{\sigma}(s)\}$, and for every $\sigma_i \in \Sigma$, let $T_i = \left\{ p_j^{\sigma_i} \mid s(p_j^{\sigma_i}) = \beta^{\sigma_i} - \mathrm{MJ}^{\sigma}(s) + jX \right\}$. From a geometric point of view, T_i consists of all the cells in σ_i, whose release time lies on the left margin of σ_i's release band. Finally, let $T = \bigcup_{\sigma_i \in \Sigma} T_i$. Assume by contradiction that for every $p_j^{\sigma} \in T$, there is no cell p_ℓ^{σ} such that $s(p_j^{\sigma}) = a(p_\ell^{\sigma})$.

Note first that in such a case, $\mathrm{MJ}^{\sigma}(s) > 0$. Otherwise, since s is aligned, for each stream σ_i the cell $p_k^{\sigma_i}$ defining β^{σ_i} satisfies both $s(p_k^{\sigma_i}) = a(p_k^{\sigma_i})$ and $s(p_k^{\sigma_i}) = \beta^{\sigma_i} - 0 + jX$.

The altered schedule s' is obtained by postponing the release of all the cells in T for some positive amount of time. As we shall prove, schedule s' is B-feasible, capacity-feasible, and has a max-jitter strictly less than $\mathrm{MJ}^{\sigma}(s)$, contradicting the optimality of s.

For each cell $p_k^{\sigma_i} \in T$ which is the j'th cell of σ (i.e, $p_k^{\sigma_i} = p_j^{\sigma}$), the exact amount of postponing time is determined by the following constraints:

1. *Avoiding buffer overflow:* Do not postpone further than the first arrival of a cell after $s(p_j^{\sigma})$. This constraint is captured by

$$\delta(p_j^{\sigma}) = \begin{cases} \min_{p_\ell^{\sigma} : a(p_\ell^{\sigma}) > s(p_j^{\sigma})} \left\{ a(p_\ell^{\sigma}) - s(p_j^{\sigma}) \right\} & \text{if } p_j^{\sigma} \text{ is not the last cell in } \sigma \\ \infty & \text{otherwise.} \end{cases}$$

2. *Maintaining FIFO order:* Recalling that $p_j^{\sigma} = p_k^{\sigma_i}$, do not postpone further than $s(p_{k+1}^{\sigma_i})$. This constraint is captured by

$$\varepsilon(p_j^{\sigma}) = \begin{cases} s(p_{k+1}^{\sigma_i}) - s(p_k^{\sigma_i}) & \text{if } p_j^{\sigma} \text{ is not the last cell in } \sigma_i \\ \infty & \text{otherwise.} \end{cases}$$

3. *Maintaining the capacity constraint:* Recalling that $p_j^\sigma = p_k^{\sigma_i}$, do not postpone further than $s(p_{k+\lambda}^{\sigma_i}) - 1$, unless $p_{k+\lambda}^{\sigma_i} \in T$. This constraint is captured by

$$\mu(p_j^\sigma) = \begin{cases} (s(p_{k+\lambda}^{\sigma_i}) - 1) - s(p_k^{\sigma_i}) & p_{k+\lambda}^{\sigma_i} \notin T \\ \infty & p_{k+\lambda}^{\sigma_i} \in T \text{ or } p_{k+\lambda}^{\sigma_i} \text{does not exist.} \end{cases}$$

Let $\delta = \min_{p_j^\sigma \in T} \delta(p_j^\sigma)$, $\varepsilon = \min_{p_j^\sigma \in T} \varepsilon(p_j^\sigma)$, and $\mu = \min_{p_j^\sigma \in T} \mu(p_j^\sigma)$, capturing the amounts of time for which these constraints are satisfied for all cells in T.

It is important to notice that δ, ε and μ are strictly greater than zero: $\delta > 0$ by its definition; $\varepsilon > 0$ since if there is a cell $p_k^{\sigma_i} \in T_i$ such that $s(p_k^{\sigma_i}) = s(p_{k+1}^{\sigma_i})$ then $s(p_{k+1}^{\sigma_i}) = \beta^{\sigma_i} - \mathrm{MJ}^\sigma(s) + kX < \beta^{\sigma_i} - \mathrm{MJ}^\sigma(s) + (k+1)X$, contradicting Lemma 3.3.5. Finally, note that by Lemma 3.3.5, and the fact that $\lambda X \geq 1$, we have for any $p_k^{\sigma_i} \in T$,

$$\begin{aligned} s(p_{k+\lambda}^{\sigma_i}) &\geq \beta^{\sigma_i} - J^{\sigma_i}(s) + (k+\lambda)X \\ &= \beta^{\sigma_i} - J^{\sigma_i}(s) + kX + \lambda X \\ &= s(p_k^{\sigma_i}) + \lambda X \\ &\geq s(p_k^{\sigma_i}) + 1. \end{aligned}$$

Since equality can only hold if $p_{k+\lambda}^{\sigma_i} \in T$, in which case $\mu(p_k^{\sigma_i}) = \infty$, we are guaranteed to have $\mu(p_k^{\sigma_i}) > 0$, which implies that $\mu > 0$.

For the purpose of analysis, define for every stream $\sigma_i \in \Sigma$,

$$\rho(\sigma_i) = \min_{p_k^{\sigma_i} \in \sigma_i \setminus T_i} \left\{ s(p_k^{\sigma_i}) - (\beta^{\sigma_i} - \mathrm{MJ}^\sigma(s) + kX) \right\}.$$

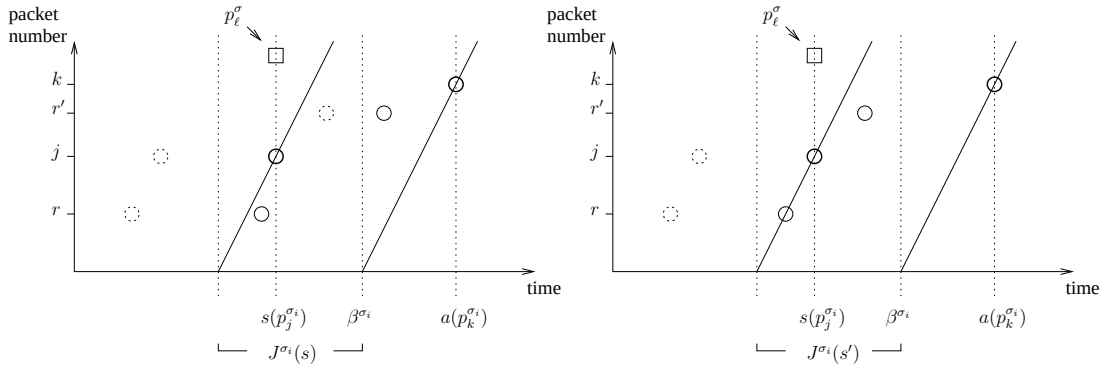
$\rho(\sigma_i)$ comes to capture how far is the rest of the stream from the left margin. Since for any $\sigma_i \in \Sigma$, $J^{\sigma_i}(s) > 0$, then $\sigma_i \setminus T_i$ is not empty and $\rho(\sigma_i) > 0$ and finite. Let $\rho = \min_{\sigma_i \in \Sigma} \rho(\sigma_i)$. It follows that $\rho > 0$ and finite.

Let $\Delta = \min\{\delta, \varepsilon, \mu, \rho\}$, and consider the following schedule that, as we shall prove, attains a jitter strictly smaller than $\mathrm{MJ}^\sigma(s)$:

$$s'(p_j^\sigma) = \begin{cases} s(p_j^\sigma) + \Delta/2 & p_j^\sigma \in T \\ s(p_j^\sigma) & \text{otherwise} \end{cases}$$

Note that this schedule is well-defined since $\Delta > 0$ and finite.

We first prove that s' is B-feasible and maintains FIFO order. Assume by way of contradiction that s' is not B-feasible, and let t be the first time the number of cells in the buffer exceeds B. By the minimality of t, there exists a cell that arrives at time t. For every cell $p_j^\sigma \in T$, no cells arrive to the buffer in the interval $[s(p_j^\sigma), s(p_j^\sigma) + \Delta/2]$ because $\Delta \leq \delta(p_j^\sigma)$, implying that t is not in any such interval. But the definition of s' yields that the content of the buffer in such a time t is the same under schedules s and s', thus contradicting the B-feasibility of s. The FIFO order of s' is maintained since $\Delta \leq \varepsilon(p_j^\sigma)$ for every $p_j^\sigma \in T$. In order to prove that s' is capacity-feasible we distinguish between two cases for each $p_k^{\sigma_i} \in T$: If $p_{k+\lambda}^{\sigma_i} \in T$ then both cells are postponed for the same duration, and therefore do not violate the capacity constraint. On the other hand, if $p_{k+\lambda}^{\sigma_i} \notin T$, capacity feasibility is maintained since $\Delta \leq \mu(p_k^{\sigma_i})$.



(a) schedule s (b) schedule s'

Figure 3.4: Outline of arrivals (dotted circles) and releases (full circles) for cells of the stream σ_i that attains the max-jitter, in an aligned release schedule, as discussed in Corollary 3.3.4 and in Lemma 3.3.7. The square represents an arrival of some cell in σ causing buffer overflow.

We conclude the proof by showing that $\mathrm{MJ}^\sigma(s') < \mathrm{MJ}^\sigma(s)$. Consider any $\sigma_i \in \Sigma$, and any $p_k^{\sigma_i}$. If $p_k^{\sigma_i} \in T$ then by the definition of s' and Lemma 3.3.5, $s'(p_k^{\sigma_i}) = s(p_k^{\sigma_i}) + \Delta/2 \geq \beta^{\sigma_i} - \mathrm{MJ}^\sigma(s) + kX + \Delta/2$. The same holds also for $p_k^{\sigma_i} \notin T$: Since $\rho(\sigma_i) \geq \Delta > \Delta/2$, it follows that $s'(p_k^{\sigma_i}) = s(p_k^{\sigma_i}) \geq \beta^{\sigma_i} - \mathrm{MJ}^\sigma(s) + kX + \rho(\sigma_i) > \beta^{\sigma_i} - \mathrm{MJ}^\sigma(s) + kX + \Delta/2$. Hence, for every $p_k^{\sigma_i}$,

$$\begin{aligned} \beta^{\sigma_i} - s'(p_k^{\sigma_i}) + kX &\leq \beta^{\sigma_i} - (\beta^{\sigma_i} - \mathrm{MJ}^\sigma(s) + kX + \Delta/2) + kX \\ &= \mathrm{MJ}^\sigma(s) - \Delta/2 \\ &< J^{\sigma_i}(s). \end{aligned}$$

By Corollary 3.3.6, $J^{\sigma_i}(s') < J^{\sigma_i}(s)$ for any stream $\sigma_i \in \Sigma$. The jitter of any other stream remains unchanged, therefore $\mathrm{MJ}^\sigma(s') < \mathrm{MJ}^\sigma(s)$, contradicting the optimality of s. □

Finally, we conclude this section by showing that there exists a polynomial-time algorithm that finds an optimal schedule for the multi-stream max-jitter problem (or returns NULL if no feasible schedule exists). Algorithm 3.1 depicts the pseudo-code of this algorithm.

Theorem 3.3.8. *There exists a polynomial-time algorithm that finds an optimal schedule for the multi-stream max-jitter problem (if a feasible schedule exists).*

Proof. Assume a feasible schedule exists. Lemma 3.3.7 implies that there is an optimal schedule s and a stream σ_i, such that $\mathrm{MJ}(s) = \beta^{\sigma_i} - a(p_l^\sigma) + kX$, for some cells $p_k^{\sigma_i} \in \sigma_i$ and $p_l^\sigma \in \sigma$. Note that for any stream σ_i, the value of β^{σ_i} can be computed in linear time using only the arrival sequence σ_i (See Algorithm 3.1, lines 18-19).

It follows that by enumerating over all possible choices of pairs $(p_k^{\sigma_i}, p_l^\sigma)$, one can find the collection of possible values of the optimal jitter (See Algorithm 3.1, function OFFLINE).

For every such value J, computing an aligned release schedule attaining jitter J that satisfies the capacity constraints, and verifying that it is B-feasible can be done in linear time by checking the feasibility of the schedule defined in Corollary 3.3.4 assuming $\mathrm{MJ}(s) = J$ (See Algorithm 3.1, functions COMPUTESCHEDULE and ISFEASIBLE). □

Algorithm 3.1 Algorithm OFFLINE

```
1: boolean function ISFEASIBLE(sequence σ, schedule s, buffer size B)
2:     BufferOccupancy ← 0
3:     σ' ← MERGE(σ, s)                          ▷ for identical values, those of s appear before those of σ
4:     for every e ∈ σ' in increasing order do
5:         if e ∈ σ then
6:             BufferOccupancy ← BufferOccupancy + 1
7:         else                                                                    ▷ i.e., e ∈ s
8:             BufferOccupancy ← BufferOccupancy − 1
9:         end if
10:        if BufferOccupancy > B then
11:            return FALSE
12:        end if
13:    end for
14:    return TRUE
15: end function

16: schedule function COMPUTESCHEDULE(sequence σ, jitter J)
17:    s ← NULL
18:    for every stream σ_i ∈ σ do
19:        β^{σ_i} ← max_j {a(p_j^{σ_i}) − jX}                          ▷ defines the right margin
20:    end for
21:    for every cell p_j^{σ_i} ∈ σ do
22:        s(p_j^{σ_i}) ← max{a(p_j^{σ_i}), β^{σ_i} − J + jX, s(p_{j−λ}^{σ_i}) + 1}   ▷ For j < λ, s(p_{j−λ}^{σ_i}) ≜ −1
23:    end for
24:    return s
25: end function

26: schedule function OFFLINE(sequence σ, buffer size B)
27:    Array of jitter values MinJitter[M]. Initially all ∞
28:    Array of schedules MinSchedule[M]. Initially all NULL
29:    for every stream σ_i ∈ σ do
30:        β^{σ_i} ← max_j {a(p_j^{σ_i}) − jX}
31:        for every p_j^{σ_i} ∈ σ_i do
32:            for every p_k^σ ∈ σ do
33:                if a(p_k^σ) ≥ a(p_j^{σ_i}) and a(p_k^σ) ≤ β^{σ_i} + jX then
34:                    α^{σ_i} ← a(p_k^σ) − jX
35:                    s ← COMPUTESCHEDULE(σ,β^{σ_i} − α^{σ_i})
36:                    if ISFEASIBLE(σ, s, B) and MinJitter[i] > β^{σ_i} − α^{σ_i} then
37:                        MinJitter[i] ← β^{σ_i} − α^{σ_i}
38:                        MinSchedule[i] ← s
39:                    end if
40:                end if
41:            end for
42:        end for
43:    end for
44:    ℓ ← arg min MinJitter
45:    return MinSchedule[ℓ]
46: end function
```

3.4 Discussion

In this chapter we examines the problem of jitter regulation and specifically, the tradeoff between the buffer size available at the regulator and the optimal jitter attainable using such a buffer. We deal with the realistic case where the regulator must handle many streams concurrently, thus answering a primary question posed in [56]. In addition we further extend the model to the case where each outgoing link is associated with a single stream, and has a bounded capacity.

We focus our attention on regulating the jitter of multiple streams with the objective of minimizing the maximum jitter attained by any of these streams. We show that the offline problem of finding a schedule that attains the optimal max-jitter can be solved in polynomial time, by a time-efficient algorithm which produces an optimal schedule. We observe that existing single-stream online algorithms can be used to devise an online algorithm for the multi-stream jitter regulation problem, at a cost of multiplying the buffer size linearly by the number of streams. We prove that such a resource augmentation is essential by providing an asymptotically matching lower bound. Our results for the online setting apply also to the problem of finding a release schedule with optimal average jitter.

Note that our offline algorithm suggests an interesting heuristic for improving the value of the jitter for an online algorithm using a buffer of size considerably less than MB, compared to the optimal jitter attainable with a buffer of size B. One can calculate an optimal schedule of a prefix of the traffic using our offline algorithm, and then prolong the schedule by attempting to send consecutive cells as equally spaced as possible. Although there are traffics in which this approach fails, as shown by our lower bound, it may prove a useful heuristic in situations where the overall traffic in the network does not change radically over time.

Sometimes, only few "misbehaved" cells significantly increase the delay jitter. Furthermore, real-life regulators may be allowed to drop cells. Therefore, it is appealing to examine the correlations between buffer size, optimal jitter, and drop ratio. In addition, it is of interest to examine situations in which the different streams share the same outgoing link; that is, the capacity constraint is imposed on the interleaving of streams rather than on each stream separately. We conjecture that this latter problem is NP-hard.

Chapter 4

Rate vs. Buffer Size - Greedy Information Gathering on the Line

4.1 Introduction

Throughput analysis of packet networks under adversarial settings has received increasing attention in recent years. A large number of works have analyzed the competitive ratios of admission and scheduling policies, measuring the throughput of the system, when traffic is given by an adversary and buffer space is limited. Such works have addressed single buffers, e.g., [4, 10, 42, 43], switches, e.g., [6, 15–17, 44], or whole networks, e.g., [5, 11, 18, 35, 41]. The adversarial setting for this investigation is motivated by theoretical interest as well as by practical needs, especially the increasing difficulty in obtaining tractable and accurate probabilistic models for network traffic. The setting of whole networks, which is especially relevant to the present work, has been studied in recent years in the framework of the *Competitive Network Throughput (CNT)* model, first introduced in [5]. This model aims at evaluating the throughput of online local-control packet admission and scheduling policies in networks with adversarial traffic, when buffer space at the routers is limited. In this model, packets are injected to various nodes over time, each with some prescribed destination and path to follow, and the goal is to maximize the overall number of packets delivered, rather than being dropped en-route due to limited buffer space. First results for this model have been obtained in [5], and were followed by additional results in, e.g., [11, 18, 35].

Most of the results mentioned above consider an arbitrary size for the buffers and a non-restricted adversary which can inject any sequence of packets into the network. They then give competitive ratios for various policies which are usually independent of the buffer size, and are a function of, e.g., the network size. This approach is clearly of merit in order to obtain results that would hold for all scenarios. However, some results, especially in the context of the throughput of single switches, lead to the question whether the size of the buffer influences the attainable competitive ratios for the problem at hand. For example, Azar and Litichevskey [15] consider the problem of scheduling a multi queue system, and present an algorithm whose competitive ratio depends on the size of the buffers, such that as the buffer size increases, the performance guarantee of the algorithm improves accordingly. In the context of the CNT model it is known

that if the buffer size is $B = 1$ then the greedy protocol (and in fact any online deterministic protocol) on the line is $\Omega(n)$ competitive [5], while if $B > 1$ better competitive ratios (such as $O(\sqrt{n})$) can be achieved by online local-control protocols [5, 11].

In this chapter we initiate a study in the framework of the CNT model of the interplay between the competitive ratio of admission and scheduling protocols, and the size of the buffers provided in the network - on one hand, and the injection rate of the traffic into the network - on the other hand. We aim at studying the question of whether providing the network with buffers whose sizes have a certain relationship with the network size and/or the injection rate of the traffic, can influence the performance of the network, measured by the competitive ratio of the deployed protocols.

As a first test case for this approach we study the topology of the line, and the problem of information gathering (i.e., all packets are destined to a single node in the network). This question received considerable attention in the literature, especially in the context of sensor networks, and wireless ad-hoc networks, e.g., [31, 47, 48], as well as in the context of the CNT model [5, 11, 18]. We give tight results, up to constant factors, for the competitive ratio of the greedy policy for information gathering on the line, as a function of the size of the line, n, the size of the buffer at each node B, and the injection rate of the adversary controlling the traffic, r. Roughly speaking, this injection rate bounds the amount of packets the adversary is allowed to inject into the network at every time step (For a formal definition of the adversary see Section 4.1.3).

Our results give insight into the question of whether provisioning the network with large buffers improves the performance of the system, measured by the throughput competitive ratio of the protocol. We show, for example, that for relatively small rates, increasing the buffer size available at the network's nodes indeed enables the greedy protocol to guarantee a better competitive ratio. However, this improvement is limited, in the sense that increasing the buffer size beyond a certain size, no longer helps in guaranteeing a better competitive ratio. Another consequence of our results is that when the adversary has rate $r \leq 1$, if buffers are sufficiently large, then the greedy protocol achieves optimal throughput, while if the buffer size is too small, then the greedy protocol cannot achieve optimal throughput. See Section 4.1.2 for a detailed description of our results.

We view our results as a first step towards a more refined analysis of throughput competitiveness, and towards providing guidelines on how should buffers be deployed in the network in adversarial settings. We believe that the results presented here give a better understanding of the role of buffer size in guaranteeing that simple protocols perform well under adversarial traffic. This may enable the use of some limited knowledge on the traffic pattern, even in an adversarial setting, which could be harnessed into providing better performance guarantees.

4.1.1 Related Work

Problems of maximizing throughput given limited size buffers and against adversarial traffic have been studied extensively in recent years e.g., [4, 6, 10, 15, 17, 42, 43]. See [28] for a short survey. These works consider the task of maximizing the number of packets transmitted from a single buffer, or from a switch, analyzing the performance of the algorithms using competitive

Range of r	Subrange of r	Result	UB	LB
$r \leq 1$	$r < \sqrt{\frac{B-1}{n}}$	Optimal	Theorem 4.2.7	
	$r \geq \sqrt{\frac{B-1}{n}}$	$\Theta\left(\max\left\{1, r\sqrt{\frac{n}{B}}\right\}\right)$	Theorem 4.2.8	Theorem 4.2.10
$1 < r < \min\{B, \sqrt{n}\}$	$r \leq \frac{n}{B}$	$\Theta\left(\sqrt{\frac{rn}{B}}\right)$	Theorem 4.3.2	Theorem 4.3.8
	$\frac{n}{B} < r < \min\{B, \sqrt{n}\}$	$\Theta(r)$	Theorem 4.3.3	Theorem 4.3.6
$r \geq \min\{B, \sqrt{n}\}$		$\Theta(\sqrt{n})$	[11]	Lemma 4.3.10

Table 4.1: Summary of results for $B \geq 2$, depending on the rate of the adversary. For every range, the UB column refers to the proof of the upper bound, and the LB column refers to the proof of the lower bound.

analysis.

In the context of whole networks, Aiello et al. [5] introduced the Competitive Network Throughput (CNT) model to study the performance of buffer management and scheduling policies which are provided with limited buffer space and against adversarial traffic. Aiello et al. show that some protocols (e.g., Nearest-to-Go (NTG)) are competitive on all networks, and some other protocols (e.g., Furthest-to-Go (FTG)) do not have bounded competitive ratio on all networks. They further show that any greedy protocol on the line is $O(n)$ competitive, that NTG is $O(n^{2/3})$ competitive, and that no greedy policy can have a competitive ratio better than $\Omega(\sqrt{n})$. These results hold for any buffer size $B > 1$. On the other hand they show that if $B = 1$, any greedy policy has competitive ratio $\Omega(n)$. Angelov et al. [11] show that for the problem of information gathering on the line (where the destination of all packets is the same node), the greedy policy is $O(\sqrt{n})$ competitive for any $B > 1$. Two works, one by Angelov et al. [11] and the other by Azar and Zachut [18], give centralized online algorithms for the throughput maximization problem on the line, with polylogarithmic competitive ratio.

The fact that there is a connection between the competitive ratio of the system and the available buffer size is suggested in a work by Azar and Litichevskey [15]. They examine the competitive ratio of online algorithms for the problem of maximizing the throughput of a system with m input ports with buffers of size B, and a single output port, where at each time step only one buffer can send a packet. They give an online algorithm with competitive ratio $\frac{e}{e-1}\left(1 + \frac{O(\log m)}{B}\right)$, which approaches $\frac{e}{e-1}$, as we provide the input ports with larger buffers. For sufficiently large buffers, this improved upon the best known previous result of 1.89 [6].

The problem of information gathering was studied in the literature under different models. For example, Kothapalli and Scheideler in [48] study this problem for the case that an adversary controls not only the injected traffic but also the activation and deactivation of network links. They give results for the line and the cycle showing tight bounds on the excess amount of buffer space that the online algorithm needs (compared to the optimal adversary) in order to deliver all injected packets.

4.1.2 Our Results

We give tight bounds on the competitive ratio of the greedy protocol for information gathering on the line. We give upper bounds and lower bound on the competitive ratios, as a function of

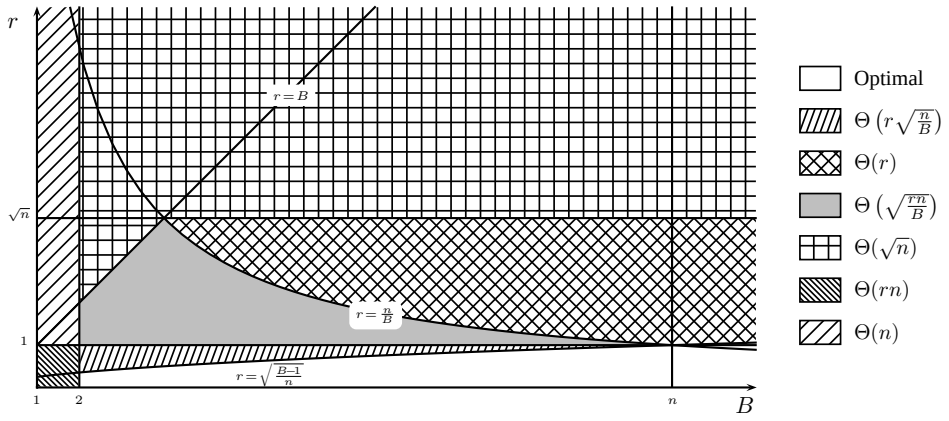


Figure 4.1: Graphic representation of results as a function of the buffer size B and the adversary's rate r. The X-axis represents the buffer size, and the Y-axis represents the adversary's rate. The different regions are marked according to the competitive ratio of the greedy policy, depending upon the pairing of buffer size and adversary rate values.

the available buffer space in every node, the rate of the adversary, and the size of the network. All our results are tight up to a constant factor.

Table 4.1 summarizes the results for the case where the buffer size is at least 2 (Section 4.4 treats the special case where $B = 1$). For different ranges of the adversary's rate, r (see Section 4.1.3 for a formal definition of the rate), it presents the competitive ratio of the greedy policy. For a graphic representation of our results, see Figure 4.1.

Note specifically that these results imply that for $r \leq 1$, if the nodes are supplied with sufficiently large buffers, then the greedy policy has optimal throughput. In addition, our results imply that for $r > 1$, increasing the buffer can help guarantee a better competitive ratio, up to a point where the competitive ratio no longer depends upon the buffer size, and becomes dependant solely of the adversary's rate.

For the case where $B = 1$, we show that if the adversary has rate $1/n < r < 1$, then the competitive ratio of the greedy protocol is $\Theta(rn)$. For $r \leq 1/n$ the greedy policy is optimal, whereas by the results in [5], for $r \geq 1$ it is $\Theta(n)$ competitive.

4.1.3 The Model

We model the network as a digraph $G = (V, E)$, $|V| = n$, $|E| = m$. The nodes in the graph represent routers and the edges represent unidirectional communication links. The system is synchronous, and time proceeds in discrete time steps. All packets in the network have equal size, and without loss of generality we assume they are of unit size. Every link has unit capacity, and can transmit at most one packet in each time step, along the direction of the link. In the tail of every link there is a buffer of size $B \geq 1$, which can store at most B packets. Packets are injected into the network, each identified by its source node, its target node, and a predesignated path which it is has to follow from source to destination. Every packet injected into the network is injected at its source node, to be stored at the output port of the first link in its path. Each time

step comprises of two substeps: the forwarding-and-injection substep followed by the switching substep. The forwarding-and-injection substep works as follows: For each link, a packet may be selected from the output buffer at the tail of the link, and this packet is forwarded to the node at the head of the link. At the same time, any number of packets can be injected into the node. In the switching sub-step packets that have arrived (or injected) to the node can be placed in the buffer of the next (or first) edge of their path. If there is not enough space in the buffer to store all packets some packets must be dropped.

A greedy protocol is a protocol that never drops a packet unless the buffer in which it has to be stored is full, and always forwards a packet from a buffer unless the buffer is empty. Protocols satisfying the latter property are sometimes referred to as *work conserving*.

We focus our attention in this work on the directed line topology. Note that in this topology, any packet is characterized solely by its source node, and target node. Furthermore, in this topology, every node has a single outgoing link. We will therefore sometime refer to a link's output buffer, as the buffer at its tail node. We further focus on the problem of *Information Gathering* on the line, where the target node of all packets is the last node of the line. In this case, every packet is characterized solely by its source node. In this work we consider greedy protocols for information gathering. Note that for this problem on the line all greedy protocols are equivalent, and we will therefore refer to *the greedy protocol* in this case. Since all greedy protocols are equivalent, unless stated differently, we assume for ease of analysis, without loss of generality, that when there is a packet arriving at a node from its preceding node, this will be the packet which is forwarded on the node's outgoing link in the next time step. We call this assumption the *en-route* assumption. Also observe that at every time step there is at least space for one new packet in any buffer, since a packet is always sent from a full buffer. We can therefore assume without loss of generality that all packets that are forwarded on a link are stored in the buffer of the node at the head of the link, and never dropped. It follows that we can assume without loss of generality that any packet accepted and stored in any source node buffer, is never dropped, i.e., packets are only dropped at injection.

We are interested in maximizing the throughput of the network, i.e., maximizing the number of packets which are delivered to the last node of the line.

We assume the injections are governed by an adversary. Given any real number r, an r-*adversary* can inject any sequence of packets as long as for every time interval of length t, at most $\lceil rt \rceil$ packets are injected into the network. Note that the adversary is allowed to inject the packets to any nodes in the network, and may well inject more than one packet simultaneously, even to the same node.

We use competitive analysis to measure the performance of the greedy protocol, under the various combinations of parameters n, B, and r. For the purpose of analysis, we assume without loss of generality that the optimal algorithm never drops a packet that it accepted at injection. Throughout the chapter, unless otherwise stated, we assume that the buffer size B is at least 2. The special case of $B = 1$ is treated in Section 4.4.

4.2 Low Rate Adversaries

4.2.1 Large Buffers

In this section we show that for any adversary of rate $r \leq 1$, if $B \geq \max\left\{2, \lceil r^2 n \rceil + 1\right\}$ then the greedy policy does not drop packets, and is therefore optimal. To this end we analyze the system as if it has unbounded buffers and no packet is dropped, and give an upper bound of $\max\left\{2, \lceil r^2 n \rceil + 1\right\}$ on the size of the buffers.

As a first step, we prove the following lemma, which bounds the overall number of packets in the network, under the greedy policy:

Lemma 4.2.1. *For any r-adversary $r \leq 1$, under the greedy policy, at any time t, the number of packets in the system is at most $\lceil rn \rceil$.*

To see this, since all greedy protocols are equivalent, it is sufficient to show that the above lemma holds for any specific greedy protocol. For analysis purposes it is convenient to consider the *Longest-in-System (LIS)* protocol. Formally, we can prove the following lemma:

Lemma 4.2.2. *LIS is equivalent to any greedy policy. Specifically, for any greedy algorithm A, any node i, and any time t, let $B_i^A(t)$ denote the number of packets in the buffer of node i at the end of time t. It then follows that for any i and t*

1. *$B_i^A(t) = B_i^{LIS}(t)$, and*
2. *LIS sends a packet from i at time t iff A sends a packet from i at time t.*

Proof. By induction on t. For $t = 0$, no packets are forwarded by any algorithm, hence any injected packet remains at its source, and the claim holds. For the induction step, consider any time $t > 0$, and any node i. By the induction hypothesis, $B_i^A(t-1) = B_i^{LIS}(t-1)$. If both buffers are empty then neither A nor LIS send a packet at time t. If the adversary injects any packets to node i in time t, then all of these packets remain in the buffer of node i under both A and LIS. If on the other hand they are not empty, then both policies send a packet in time t, and after having sent a packet, the equality is still maintained. Any packets injected by the adversary to node i in time t are then added to the packets already in the buffer under both policies, and equality is kept. □

The following lemma enables us to give a bound on the amount of time every packet stays in the network:

Lemma 4.2.3. *Under LIS, for any adversary of rate $r \leq 1$, consider any packet p injected to node i in time t. Then for every $j \geq i$, p arrives to node j by time $t + j$, and p is sent from node j by time $t + j + 1$.*

Proof. By induction on t. For $t = 0$, since $r \leq 1$, p is the only packet injected in time t, which implies it is the first packet injected. Due to the LIS policy, it therefore traverses the network without being delayed at any node. Hence, for every $j \geq i$, it arrives to node j at time $t + (j - i) \leq t + j$, and leaves in the next time step, i.e. in time $t + (j - i) + 1 \leq t + j + 1$.

For the induction step on t, we prove the result by induction on j. For $j = i$, the first part of the claim trivially holds. By the induction hypothesis on t, every packet q injected to any node $v \leq j$ in time $t' < t$ would leave node j by time $t' + j + 1 \leq t + j$, hence LIS would schedule p to be sent from node j by time $t + j + 1$.

For the induction step, consider any $j > i$. By the induction hypothesis on $j - 1$, p is sent from node $j - 1$ by time $t + (j - 1) + 1 = t + j$, and therefore arrives to node j by time $t + j$. By the induction hypothesis on t, every packet q injected to any node $v \leq j$ in time $t' < t$ would leave node j by time $t' + j + 1 \leq t + j$, hence LIS would schedule p to be sent from node j by time $t + j + 1$. □

Applying the above lemma with $j = n$ we obtain the following corollary:

Corollary 4.2.4. *Under LIS, for any adversary of rate $r \leq 1$, every packet is in the system for at most n time units.*

The following lemma gives a bound on the overall number of packets in the network at any given time under the LIS protocol.

Lemma 4.2.5. *Under LIS, for any adversary of rate $r \leq 1$, and at any time t, the number of packets in the system in time t is at most $\lceil rn \rceil$.*

Proof. Consider any time t. By Corollary 4.2.4, any packet injected to the system before time $t - n$ has already been delivered. Hence the system holds only packets injected during the interval $(t - n, t]$. By the definition of the adversary, the maximum number of packets injected during such an interval is at most $\lceil rn \rceil$. □

Combining the above lemma with the fact that all greedy policies are equivalent, as shown in Lemma 4.2.2, we conclude the proof of Lemma 4.2.1. Note that Lemma 4.2.1 guarantees that for any r-adversary such that $r \leq 1$, if $B \geq \lceil rn \rceil$ then any greedy policy does not drop packets, and hence any greedy policy is optimal. In what follows we show that the same result holds even for buffers of smaller size.

Lemma 4.2.6. *For any adversary of rate $r \leq 1$, and any greedy policy, at any time t there are at most $\max\left\{2, \lceil r^2 n \rceil + 1\right\}$ packets in every buffer.*

Proof. Let i be any node in the system. If at the beginning of time step t there are more than 2 packets at i's buffer, then at time $t - 1$ the node was not empty (since at most 2 packets can arrive in each time step). Let t' be the latest time prior to t where the buffer is empty. Without loss of generality assume $t' = 0$. We distinguish between two cases:

Case 1: $t \leq \lceil nr \rceil$:

In this case, at time t the number of packets in node i is at most

$$t + \lceil tr \rceil - t = \lceil tr \rceil \leq \lceil r^2 n \rceil + 1.$$

Case 2: $t > \lceil nr \rceil$:

Let $\varepsilon = \lceil tr \rceil - tr$, and note that $0 \leq \varepsilon < 1$. At time t the number of packets in node i is at most

$$\begin{aligned} \lceil nr \rceil + \lceil tr \rceil - t &= \lceil nr \rceil + t(r-1) + \varepsilon \\ &= \lceil nr \rceil - t(1-r) + \varepsilon \\ &< \lceil nr \rceil - \lceil nr \rceil (1-r) + \varepsilon \\ &\leq \lceil nr \rceil r + 1, \end{aligned}$$

where the first term follows from Lemma 4.2.1, and the inequality follows from the fact that $r \leq 1$. Since any integer m which satisfies $m < \lceil rn \rceil r$ also satisfies $m \leq \lceil r^2 n \rceil$, it follows that the number of packets in node i is at most $\lceil r^2 n \rceil + 1$. □

The following theorem is an immediate consequence of the above lemma:

Theorem 4.2.7. *For any r-adversary such that $r \leq 1$, if $B \geq \max\left\{2, \lceil r^2 n \rceil + 1\right\}$ then the greedy policy does not drop packets, and thus is optimal.*

4.2.2 Small Buffers

In this section we give tight bounds on the competitive ratio of the greedy policy against any r-adversary with $r \leq 1$, in a network which is supplied with relatively small buffers. Specifically, we show that if $2 \leq B \leq \Theta(r^2 n)$, then the greedy policy has competitive ratio $\Theta(r\sqrt{\frac{n}{B}})$.

Upper Bound

Theorem 4.2.8. *If $2 \leq B \leq n$, and the packets are injected by an r-adversary with $\sqrt{\frac{B-1}{n}} \leq r \leq 1$, then the greedy policy is $O(\max\left\{1, r\sqrt{\frac{n}{B}}\right\})$ competitive.*

Proof. For the purpose of the analysis we divide time into a sequence of intervals $P_0, K_0, P_1, K_1, P_2, K_2, \ldots$. Intervals P are defined by the number of packets that the adversary accepts. Intervals K will be fixed length intervals of length k. Formally, let $P_i = [s_i, t_i + 1)$, and $K_i = [t_i + 1, u_i)$ where

1. $s_0 = 0$ and for $i > 1$, $s_i = u_{i-1}$,
2. t_i is the earliest time after s_i where the adversary accepts $3 \cdot \lceil rn \rceil$ packets during the interval $[s_i, t_i + 1)$, and
3. $u_i = t_i + k$, for $k = \Theta(n)$.

We start by showing that we can identify $\Omega(\min\left\{rn, \sqrt{nB}\right\})$ distinct packets residing in the buffers of the greedy policy during P_i.

If the greedy policy accepts at least $\lceil rn \rceil$ of the new packets injected by the adversary during P_i then we have at least $\Omega(rn)$ packets residing in the buffers of the greedy policy during P_i.

Assume now that the greedy policy does not accept at least $\lceil rn \rceil$ of the new packets injected by the adversary during P_i. It follows that the greedy policy drops during P_i at least $2 \cdot \lceil rn \rceil$ packets.

We say that a node j is *bad in P_i* if at least one packet was dropped in j during P_i. Note that if a packet is dropped in j at time t, then the buffer at that node is full at that time, and furthermore, due to the en-route assumption, at least $B - 1 \geq \frac{B}{2}$ of the packets residing in

node j at time t have been injected to j itself. Let x denote the number of bad nodes in P_i. If $x \geq \sqrt{\frac{n}{B}}$, then we can identify at least $x\frac{B}{2} = \Omega(\sqrt{nB})$ distinct packets residing in the buffers of the greedy policy during P_i. Assume now that $x < \sqrt{\frac{n}{B}}$. Recall that the greedy policy has dropped at least $2 \cdot \lceil rn \rceil$ packets during P_i, hence in at least one of the bad nodes the greedy policy has dropped at least $\frac{2 \cdot \lceil rn \rceil}{\sqrt{\frac{n}{B}}} \geq 2r\sqrt{nB}$ packets. Observe that by the assumption that $r \geq \sqrt{\frac{B-1}{n}}$ we are guaranteed to have $2r\sqrt{nB} \geq 2$. We now use the following lemma, whose proof appears later in the sequel.

Lemma 4.2.9. *Any bad node j such that at least $q \geq 2$ packets were dropped at j during P_i, forwards $\Omega(q \cdot \frac{1}{r})$ packets during P_i.*

It follows that there is at least one bad node from which

$$\Omega\left(2r\sqrt{nB} \cdot \frac{1}{r}\right) = \Omega(\sqrt{nB})$$

packets have been forwarded during P_i, which means that we can identify $\Omega(\sqrt{nB})$ distinct packets residing in the buffers of greedy during P_i.

We can therefore conclude that there are $\Omega(\min\left\{rn, \sqrt{nB}\right\})$ distinct packets residing in the buffers of the greedy policy during P_i.

By the fact that $B \leq n$, we have

$$k = \Theta(n) = \Omega(n + \min\left\{rn, \sqrt{nB}\right\}).$$

Since we have shown that at least $\Omega(\min\left\{rn, \sqrt{nB}\right\})$ distinct packets resided in the buffers under the greedy policy during P_i, it follows that at least this number of packets were delivered by the greedy protocol during $P_i \cup K_i$.

As to the adversary, note that by the choice of k - the length of interval K_i - the overall number of packets accepted by the adversary during $P_i \cup K_i$ is bounded by $3 \cdot \lceil rn \rceil + r \cdot \Theta(n) = O(rn)$.

Summing the above over all i, we obtain a lower bound of

$$\Omega(\min\left\{jrn, j\sqrt{nB}\right\})$$

on the number of packets delivered by the greedy policy by the end of K_j, where on the other hand the same summation yields an upper bound of $O(jrn)$ on the number of packets accepted by the adversary by the end of interval K_j, which clearly also bounds the number of packets delivered by the adversary by the end of K_j. It therefore follows that the ratio between the number of packets delivered by the adversary and the number of packets delivered by the greedy policy is $O(\max\left\{1, r\sqrt{\frac{n}{B}}\right\})$, which completes the proof. □

Proof of Lemma 4.2.9. By the assumption, we know that at least 2 packets were dropped at node j. Consider any two consecutive events in which a packet was dropped at node j, and assume without loss of generality that the first drop was at time 0, and the second drop was at time t. Note that for every node j' and time s, a packet is dropped at the switching substep of time s only when there has been both an injection into node j' and a forwarding to node j' in

the forwarding-and-injection substep of time s, and the buffer of j' is full at the beginning of the forwarding-and-injection substep. Furthermore, since every two consecutive injections are at least $\lfloor 1/r \rfloor \geq 1$ time apart, we necessarily have $t > 0$. If the buffer at node j is full during the entire interval $[0, t]$, then clearly at least $t \geq \lfloor 1/r \rfloor = \Omega(1/r)$ packets have been forwarded from node j under the greedy policy. Otherwise, let $0 < s < t$ be the last time prior to t in which the buffer at node j was not full at the end of time slot s. By the maximality of s, and the fact that $r \leq 1$, it follows that at the end of time s there were $B - 1$ packets in the buffer of node j, and at the forwarding-and-injection substep of time $s + 1$ one packet arrived to node j on its incoming link, and one packet was injected to node j. By the fact that inter-injection time is at least $\Omega(1/r)$, it follows that the interval $(s, t]$ is of length at least $\Omega(1/r)$, and since the buffer was always full during this interval, it follows that one packet was forwarded from node j in every time step in this interval, i.e., at least $\Omega(1/r)$ packets were forwarded from node j in the interval $[0, t]$.

Since this holds for every two consecutive events of packets being dropped at j, and by the assumption on j there were at least $q \geq 2$ packets dropped at j during P_i, we conclude that at least $\Omega(q \cdot \frac{1}{r})$ packets were forwarded from node j during P_i. □

Lower Bound

In this section we prove that the upper bound given in Theorem 4.2.8 is tight up to a constant factor, for buffers smaller than $O(r^2 n)$. Note that for any constant $0 < c < 1$, and any r-adversary such that $r \leq 1$, if $cr^2 n \leq B \leq \lceil r^2 n \rceil$ then Theorem 4.2.8 guarantees that the greedy policy is $O(1)$-competitive. Therefore it is enough to prove our lower bound for buffers of size less than $\frac{1}{16} r^2 n$.

Theorem 4.2.10. *For any $r \leq 1$, if $2 \leq B < \frac{1}{16} r^2 n$, then there exists an r-adversary A such that the ratio between the throughput of A and that of the greedy policy is $\Omega\left(r\sqrt{\frac{n}{B}}\right)$.*

Proof. The adversary will inject packets in two epochs. We will consider the line as divided into two blocks, where the second block is divided into segments. In the first epoch the adversary injects only to the first block, whereas in the second epoch the adversary injects only to the second block. The goal of the injection sequence in the first epoch is to generate a continuous sequence of packets arriving at the second block. The second block is divided into segments, where the injection during the second epoch will cause the greedy policy to drop packets in every segment. As the analysis will show, the overall number of packets accepted by the greedy policy would be proportional to the injections made to the first block, whereas the adversary can accept all the packets injected.

Formally, let $r' = \frac{1}{\lceil 1/r \rceil}$. It follows that $r/2 \leq r' \leq r$, and $1/r'$ is integral. Let $d = \lfloor \sqrt{nB} \rfloor$. Consider the line as composed of two blocks of nodes, where the first block consists of the nodes $0, \ldots, \frac{d}{r'} - 1$, and the second block consists of nodes $\frac{d}{r'}, \ldots, n$. We divide the second block into $k = \lfloor \frac{n}{d} \rfloor - \frac{1}{r'}$ segments of length d each, $S_0, \ldots, S_{k-1}$.

Note that by the assumption on B and the choice of r' and d, the number of nodes in the first block is at most

$$\frac{d}{r'} = \frac{\lfloor \sqrt{nB} \rfloor}{r'} \leq \frac{\sqrt{nB}}{r/2} < 2\frac{\sqrt{r^2 n^2/16}}{r} = \frac{n}{2}.$$

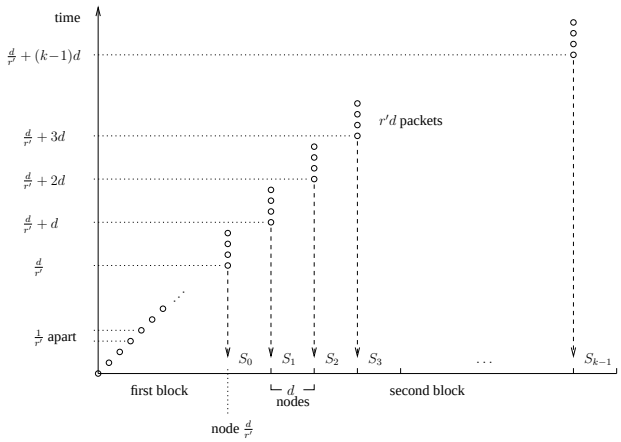


Figure 4.2: Outline of the injection pattern for the adversary showing the $\Omega\left(r\sqrt{\frac{n}{B}}\right)$ lower bound. The X-axis represents the line network, and each circle represents the injection of a packet. Out of the $r'd$ packets injected to every segment in the second block, only B packets would be absorbed by the greedy policy.

Since there remain at least $\frac{n}{2}$ nodes in the second block, and the length of every segment in the second block is

$$d = \lfloor \sqrt{nB} \rfloor \leq \sqrt{nB} < \sqrt{\frac{r^2n^2}{16}} = \frac{rn}{4} \leq \frac{n}{4},$$

we are guaranteed to have at least two segments in the second block.

The injection sequence of the adversary is divided into two epochs, as follows:

Epoch 1: For every $i = 0, \ldots, d-1$, inject a packet to node $\frac{i}{r'}$ in time $\frac{i}{r'}$.

Epoch 2: For every segment $j = 0, \ldots, k-1$, inject $\lfloor r'd \rfloor$ packet to the first node of S_j, one every $1/r'$ time units, starting from time $\frac{d}{r'} + jd$. Note that by the choice of r, r' and d we have $\lfloor r'd \rfloor \geq 2$.

See Figure 4.2 for an outline of the injection sequence.

In addition, note that since the above injection sequence does not inject more than one packet every $1/r'$ time units, the injection rate is at most $r' \leq r$, hence it corresponds to an r-adversary.

We now turn to analyze the performance of the greedy policy given the above injection sequence. First note that the greedy policy accepts all the packets injected during epoch 1. To see this, notice that the adversary injects at most one packet to every node. It follows that there is at most one time unit where the node receives two packets simultaneously - one from its preceding node, and one injected by the adversary. Since by our assumption $B \geq 2$, the greedy policy does not drop packets during epoch 1.

The following lemma, whose proof appears in the sequel, shows that starting from time $\frac{d}{r'}$, there is a continuous sequence of d packets arriving to the first node of S_0 from its preceding node.

Lemma 4.2.11. *For every $i = 0, \ldots, d-1$, there is a continuous sequence of $i+1$ packets leaving node $\frac{i}{r'}$, starting from time $\frac{i}{r'} + 1$.*

The following lemma, whose proof appears later in the sequel, bounds the number of packets which leave any of the segments in the second block, under the assumption that $2 \leq B < \frac{1}{16}r^2n$:

Lemma 4.2.12. *For every $i = 0, \ldots, k-1$, there is a continuous sequence of $d + (i+1)B$ packets leaving S_i, entering segment S_{i+1} as of time $\frac{d}{r'} + (i+1)d$.*

Since the number of segments in the second block is

$$\lfloor \frac{n}{d} \rfloor - \frac{1}{r'} = \lfloor \frac{n}{\lfloor \sqrt{nB} \rfloor} \rfloor - \frac{1}{r'} = O\left(\sqrt{\frac{n}{B}}\right),$$

by Lemma 4.2.12, the number of packets delivered by the greedy policy is $O(d + \sqrt{\frac{n}{B}}B) = O(\sqrt{nB})$.

The adversary injects at least

$$d + \left(\lfloor \frac{n}{d} \rfloor - \frac{1}{r'}\right) r'd = \Theta(r'n)$$

packets. It can keep them all by not forwarding packets in the first block, and spreading the $r'd$ packets injected to segment S_i throughout the segment while not sending packets between different segments. Therefore after a flush-phase at the end of the injection sequence, the adversary can deliver all the packets it has accepted.

It follows that the ratio between the number of packets delivered by the adversary and the number of packets delivered by the greedy policy is at least

$$\frac{\Theta(r'n)}{O(\sqrt{nB})} = \Omega\left(r\sqrt{\frac{n}{B}}\right),$$

which completes the proof of Theorem 4.2.10. □

Proof of Lemma 4.2.11. The proof is by induction on i. For the base case of $i = 0$, by the definition of the greedy policy and the adversary, there is one packet leaving node 0, starting at time 1.

For the inductive step, assume the claim holds for i. It follows that there is a continuous sequence of $i+1$ packets leaving node $\frac{i}{r'}$, starting from time $\frac{i}{r'} + 1$. During the time interval $(\frac{i}{r'}, \frac{i+1}{r'} - 1]$, no injections are made by the adversary, and by the end of this interval, the head of the sequence has arrived to node $\frac{i+1}{r'} - 1$. In time $\frac{i+1}{r'}$ the head of the sequence arrives to node $\frac{i+1}{r'}$, and at the same time a packet is injected to node $\frac{i+1}{r'}$. Due to the en-route assumption, the injected packet will be stored in the buffer until the entire sequence of $i+1$ packets has traversed the node. Note that the first packet in the sequence will leave node $\frac{i+1}{r'}$ one time unit after its arrival, i.e., in time $\frac{i+1}{r'} + 1$. After the last packet of the sequence entering the node, leaves the node, the packet injected to the node 'joins' the sequence, thus prolonging it to a continuous sequence of $i + 2 = (i+1) + 1$ packets which has started leaving node $\frac{i+1}{r'}$ in time $\frac{i+1}{r'} + 1$. This completes the proof of the lemma. □

Proof of Lemma 4.2.12. The proof is by induction on i. For the base case, note that by Lemma 4.2.11, there is a continuous sequence of d packets entering the first node of S_0, starting from time $\frac{d}{r'}$. It follows that during d time units, there is a packet arriving to the first node of S_0

from its preceding node. In addition, during these d time units, there are $\lfloor r'd \rfloor$ packets injected by the adversary to the first node of S_0. Due to the en-route assumption, none of these packets are forwarded from this node until the entire sequence of d packets arriving on the incoming link has ended. Note that by our assumption that $2 \leq B < \frac{1}{16}r^2 n$, we obtain that $r > 4\sqrt{\frac{B}{n}}$. It follows that

$$\begin{aligned} \lfloor r'd \rfloor &\geq \lfloor \frac{r\lfloor \sqrt{nB} \rfloor}{2} \rfloor \\ &\geq \lfloor 2\frac{\sqrt{B}\lfloor \sqrt{nB} \rfloor}{\sqrt{n}} \rfloor \\ &\geq \lfloor 2\frac{\sqrt{B}(\sqrt{nB}-1)}{\sqrt{n}} \rfloor \\ &= \lfloor 2\left(B - \sqrt{\frac{B}{n}}\right) \rfloor > B, \end{aligned}$$

where the last inequality follows from the fact that in our case $2 \leq B < \frac{1}{16}r^2 n \leq \frac{n}{16}$. The node can store only B out of these $\lfloor r'd \rfloor$ injected packets, which are then forwarded immediately after the sequence arriving on the incoming link has terminated. This prolongs the sequence leaving the first node of S_0 by additional B packets, to a total of $d + B = d + (0+1)B$ packets, which start leaving the first node of S_0 in time $\frac{d}{r'} + 1$. Since the length of S_0 is d nodes, this sequence enters segment S_1 as of time $\frac{d}{r'} + d = \frac{d}{r'} + (0+1)d$. This completes the base case.

For the inductive step, assume the claim holds for i. It follows that there is a continuous sequence of $d + (i+1)B$ packets leaving S_i, entering segment S_{i+1} as of time $\frac{d}{r'} + (i+1)d$. Starting from this time, during a period of d time units, the adversary injects $r'd$ packets to the first node of S_{i+1}. Similar to the base case, due to the en-route assumption, none of these packets are forwarded from this node until the entire sequence of $d + (i+1)B$ packets arriving on the incoming link has ended. Since the node can only store B out of the $\lfloor r'd \rfloor$ packets injected by the adversary, these packets 'join' the sequence arriving on the incoming link, thus the continuous sequence of packets leaving the node comprises of $d + (i+1)B + B = d + (i+2)B$ packets. By the fact that the length of S_{i+1} is d, this sequence starts entering segment S_{i+2} as of time $\frac{d}{r'} + (i+1)d + d = \frac{d}{r'} + (i+2)d$, which completes the proof of the lemma. □

4.3 High Rate Adversaries

In this section we treat the case of adversaries of high rates, i.e., of rates $r > 1$. We give tight bounds on the competitive ratios obtained by the greedy policy in this case. These bounds are a function of the network size n, the buffer size B, and the injection rate r. Interestingly, different functions apply for different combinations of these values.

4.3.1 Upper Bounds

Let $M = \max\{n, B\}$. The following lemma shows an upper bound in terms of M on the performance of the greedy policy, against any r-adversary with $r > 1$

Lemma 4.3.1. *For any r-adversary such that $r > 1$ the greedy policy is $O\left(\sqrt{\frac{rM}{B}} + r\right)$ competitive.*

Proof. The following proof is an extension of the proof appearing in [11].

For the purpose of the analysis we divide time into a sequence of intervals $P_0, K_0, P_1, K_1, P_2, K_2, \ldots$. Intervals P are defined by the number of packets that the adversary accepts. Intervals K will be fixed length intervals of length k. Formally, let $P_i = [s_i, t_i + 1)$, and $K_i = [t_i + 1, u_i)$ where

1. $s_0 = 0$ and for $i > 1$, $s_i = u_{i-1}$,
2. t_i is the earliest time after s_i where the adversary accepts $\lceil rM \rceil$ packets during the interval $[s_i, t_i + 1)$, and
3. $u_i = t_i + k$, for $k = \Theta(\sqrt{rMB} + n)$.

In what follows, we compare the throughput of the adversary and the throughput of the greedy algorithm in every interval $P_i \cup K_i$.

We start by showing that we can identify $\Omega(\sqrt{rMB})$ distinct packets residing in the buffers of the greedy policy during P_i.

Note first that if the greedy policy accepts at least $\frac{rM}{2}$ of the packets accepted by the adversary during P_i, then since $rM \geq \sqrt{rMB}$ for $r \geq 1$, we are guaranteed to have $\Omega(\sqrt{rMB})$ packets residing in the buffers of the greedy policy during P_i.

Assume next that the greedy policy does not accept at least $\frac{rM}{2}$ of the new packets accepted by the adversary. It follows that it drops at least $\frac{rM}{2}$ of the new packets accepted by the adversary during P_i. For the purpose of the proof we define a dynamic weight assignment to packets stored by the greedy protocol.

Initializing the weights: Every packet accepted by the greedy policy has its weight initialized to zero in the moment of its injection, and all packets not yet delivered have their weight reset to zero in the beginning of any interval P_i.

Increasing the weights: Any interval P_i is divided into periods for every node separately. The k'th period of a node is defined by the time interval $[x_k, x_k + B)$, where x_k is the earliest time a packet is dropped from the node after the end of the previous period. In the beginning of every period, we increase the weight of every packet in the node's buffer by 2. There is no weight increase during the intervals K_i.

Note that a packet is dropped at node i at the beginning of a period iff the buffer is full at this time, i.e., there are B packets in the buffer. By increasing the weight of each of these packets by 2, the overall weight increase is $2B$, which is an upper bound on the number of packets the adversary may accept into node i, and the greedy policy lose, during this period (of length B).

We now show that during interval P_i the greedy policy stored in its buffers at least $\Omega(\sqrt{rMB})$ distinct packets. Let $2c$ be the maximum weight a packet has at the end of interval P_i, where c is some positive integer.

By the fact that for every node j, the weight increase in every period of node j is an upper bound on the number of packets accepted by the adversary and dropped by the greedy protocol during the period at node j, and since the number of packets that were dropped by the greedy protocol but accepted by the adversary during P_i is at least $\frac{rM}{2}$, we have that the total weight of the packets of greedy is at least $\frac{rM}{2}$, and we can therefore identify at least $\frac{rM}{2c} = \Omega(\frac{rM}{c})$ distinct packets residing in the buffers of greedy during P_i. It follows that if $c = 1$, this is at least $\Omega(rM) = \Omega(\sqrt{rMB})$, and we are done.

Assume next that $c \geq 2$, and let p be any packet with weight $2c$. Note that p may have already been delivered by the algorithm. The weight of p can be divided into two categories, such that $2c = 2w + 2v$:

Weight given at p's origin node: Denote it by $2w$. It follows that p spent at least $B(w-1)$ time units at its origin node, since it was there during w periods, each lasting B time units. Since the algorithm is greedy, in every such time unit, one packet was sent from the origin node, i.e. at least $B(w-1)$ packets were sent during these time units. These are all different packets, which p will never 'beat' to the end, due to our en-route assumption.

Weight given at p's transit nodes: Denote it by $2v$. In every transit node where p had its weight increased, there are $B-1$ packets left behind (because the weight is increased only in time of overflow, where the buffer is full). Since p moves continuously, the sets of packets in two distinct such transit nodes are disjoint, because of the en-route assumption. Therefore, there are at least $v(B-1)$ different packets left 'behind' p.

The number of packets stored by greedy during P_i is at least

$$\begin{aligned} 1 + B(w-1) + v(B-1) &= cB - B - v + 1 \\ &\geq cB - B - c + 1 \\ &= (c-1)(B-1) = \Omega(cB). \end{aligned}$$

It follows that if the algorithm dropped at least $\frac{rM}{2}$ of the packets accepted by the adversary during P_i, it had stored in its buffers at least

$$\Omega\left(\max\left\{cB, \frac{rM}{c}\right\}\right) = \Omega(\sqrt{rMB})$$

packets during interval P_i.

We can therefore conclude that in any case there were at least $\Omega(\sqrt{rMB})$ packets residing in the buffers under the greedy policy during P_i.

When considering the adversary, note that by the choice of k - the length of interval K_i - the overall number of packets accepted by the adversary during the interval $P_i \cup K_i$ is upper bounded by $\lceil rM \rceil + r \cdot \Theta(\sqrt{rMB} + n) = O(rM + r\sqrt{rMB})$.

Furthermore, since we have shown that $\Omega(\sqrt{rMB})$ distinct packets resided in the buffers under the greedy policy during P_i, and since $k = \Theta(\sqrt{rMB} + n)$, it follows that at least $\Omega(\sqrt{rMB})$ packets were delivered during $P_i \cup K_i$ under the greedy policy.

By summing the above over all i, we obtain a lower bound of $\Omega(j\sqrt{rMB})$ on the number of packets delivered by the greedy policy by the end of K_j, where on the other hand the same summation yields an upper bound of $O(j(rM + r\sqrt{rMB}))$ on the number of packets accepted by the adversary by the end of interval K_j, which clearly also bounds the number of packets delivered by the adversary by the end of K_j. It therefore follows that the ratio between the number of packets delivered by the adversary and the number of packets delivered by the greedy policy is $\left(\frac{O(rM+r\sqrt{rMB})}{\Omega(\sqrt{rMB})}\right) = O\left(\sqrt{\frac{rM}{B}} + r\right)$, which completes the proof. □

The above lemma implies two upper bounds on the performance of the greedy policy, depending on the rate of the adversary. The first applies to adversaries with rates bounded by $\frac{n}{B}$:

Theorem 4.3.2. *For any r-adversary such that $1 < r \leq \frac{n}{B}$, the greedy policy is $O(\sqrt{\frac{rn}{B}})$ competitive.*

Proof. Assume $r > 1$ also satisfies $1 < r \leq \frac{n}{B}$. In particular in this case, we have $B < n$, which implies $M = n$. By the assumption that $1 < r \leq \frac{n}{B}$, we have $r \leq \sqrt{\frac{rn}{B}} = \sqrt{\frac{rM}{B}}$. It therefore follows by Lemma 4.3.1 that the competitive ratio is at most $O\left(\sqrt{\frac{rn}{B}}\right)$. □

The following theorem gives an upper bound for the remaining range of r.

Theorem 4.3.3. *For any r-adversary such that $r > 1$ and $r > \frac{n}{B}$, the greedy policy is $O(r)$ competitive.*

Proof. Assume $r > 1$ also satisfies $r > \frac{n}{B}$. If $B \geq n$ then $\frac{rM}{B} = r$, hence by Lemma 4.3.1, the competitive ratio is $O(\sqrt{r} + r) = O(r)$. If on the other hand $B < n$, then by the assumption that $r > \frac{n}{B}$, we have $\frac{rM}{B} = \frac{rn}{B} < r^2$. It therefore follows by Lemma 4.3.1 that the competitive ratio is at most $O(\sqrt{r^2} + r) = O(r)$. □

Angelov et al. [11] have shown that for all r, and regardless of the buffer size B, the greedy policy is $O(\sqrt{n})$ competitive. Combining their result with Theorems 4.3.2 and 4.3.3, we obtain the following two corollaries:

Corollary 4.3.4. *For any r-adversary such that $1 < r \leq \frac{n}{B}$, the greedy policy is $\min\left\{O(\sqrt{\frac{rn}{B}}), O(\sqrt{n})\right\}$ competitive.*

Corollary 4.3.5. *For any r-adversary such that $r > 1$ and $r > \frac{n}{B}$, the greedy policy is $\min\{O(r), O(\sqrt{n})\}$ competitive.*

4.3.2 Lower Bounds

In this section we present two lower bounds which combined with the upper bounds presented in Section 4.3.1, enable us to characterize the performance of the greedy policy, up to a constant factor, for any r-adversary such that $r > 1$.

Theorem 4.3.6. *For any $4 < r < \sqrt{n}$, and for any buffer size B, there exists an r-adversary A such that the ratio between the throughput of A and that of the greedy policy is $\Omega(r)$.*

Proof. We consider the line as divided into segments, and have the adversary inject at most one packet in every time step to every segment. Given any rate $4 < r < \sqrt{n}$, we show that the number of segments is at most r, hence the injection corresponds to an r-adversary. As the analysis will show, the overall number of packets accepted by the greedy policy would be proportional to the injections made to the last segment, whereas the adversary can accept all the packets injected.

Formally, Let $4 < r < \sqrt{n}$, and let $d = \lceil \frac{n}{r^2} \rceil$. Consider the line as composed of $k = \lfloor \sqrt{\frac{n}{d}} \rfloor$ segments $S_0, \ldots, S_{k-1}$, such that the length of segment S_i is $(i+1)d$.

Note that by the assumption on r we have $2 \leq d \leq \lceil \frac{n}{16} \rceil$, and the overall length of the segments is $\sum_{i=1}^{k} id = \frac{k(k+1)d}{2} \leq k^2 d \leq n$.

We now describe the sequence of injections generated by an r-adversary A. For every $i = 0, \ldots, k-1$, A injects $(i+1)dB$ packets to the first node of segment S_i, starting at time $t_i = \sum_{j=0}^{i} jd$.

See Figure 4.3 for an outline of the injection sequence.

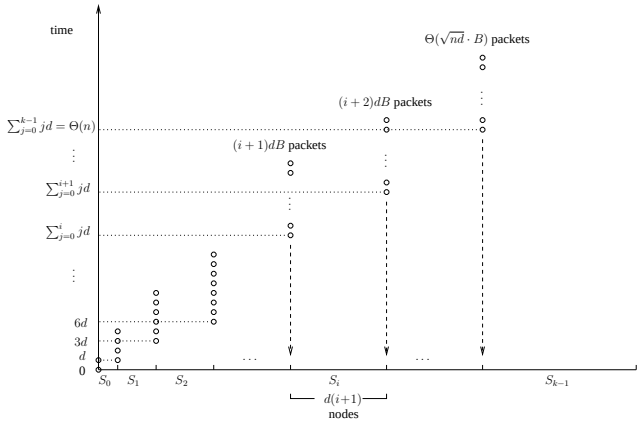


Figure 4.3: Outline of the injection pattern for the adversary showing the $\Omega(r)$ lower bound. The X-axis represents the line network, and each circle represents the injection of a packet. In every segment S_i except for S_0, out of the $(i+1)dB$ packets injected, only B packets would be absorbed by the greedy policy.

First note that by the choice of k, we have $k \leq \sqrt{\frac{n}{d}} \leq r$. Since the adversary injects at most one packet to every segment in every time step, we are guaranteed that the above injection sequence corresponds to an r-adversary. Furthermore, since $d \leq \lceil \frac{n}{16} \rceil$ we have that $k \geq 2$.

The following lemma, whose proof appears in the sequel below, enables us to bound the number of packets leaving every segment S_i under the greedy policy.

Lemma 4.3.7. *Under the greedy policy, for any $i \geq 1$, the packets leaving segment S_i form a continuous sequence of $(i+1)dB + B$ packets, which start arriving to S_{i+1} in time t_{i+1}.*

It follows that the greedy policy delivers $O(kdB) = O(\sqrt{dn} \cdot B) = O\left(\frac{nB}{r}\right)$ packets.

The number of packets injected to the network by the adversary is nB, and the adversary may successfully deliver them all by storing the $(i+1)dB$ packets injected to segment S_i in the buffers of that segment, and not forwarding any packet across different segments until the injection sequence has terminated.

It follows that the ratio between the number of packets delivered by the adversary and the number of packets delivered by the greedy policy is $\Omega(r)$. This concludes the proof of the theorem. □

Proof of Lemma 4.3.7. The proof is by induction on i. For the base case clearly the sequence of packets leaving S_0 forms a continuous sequence of dB packets, which by the choice of the length of S_0, starts arriving to the first node of segment S_1 in time d. By the definition of the adversary, starting at time $t_1 = d$, there is a sequence of $2dB$ packets which are injected to the first node of segment S_1, one every time step. It follows that during dB time units, starting from

time t_1, there are two packets arriving to the first node of segment S_1 - one on the incoming link, and one injected by the adversary. By the en-route assumption, none of the packets injected by the adversary during these dB time units are forwarded from this node until the entire sequence of packets arriving on the incoming link has ended. Since $d \geq 2$, $dB > B$, and therefore the greedy policy can store only B of the packets injected to the first node of S_1 during these dB time steps, and must drop the remaining packets. In the following dB time steps, there is only one packet arriving to the first node of segment S_1 - the packet injected by the adversary. The first node of segment S_1 therefore forwards one packet in every time step during these dB time units as well, and maintains a full buffer during all this time. In the following B time units, the first node of segment S_1 flushes its buffer, and forwards one packet in every time step. The packets forwarded by the first node of segment S_1 therefore form a continuous sequence of $dB + dB + B = 2dB + B = (i+1)dB + B$ packets, which by the definition of the length of S_1, start arriving to the first node of S_2 in time $t_1 + 2d = 3d = t_2$. This completes the base case.

For the inductive step, assume the claim holds for $i \geq 1$. It follows that there is a continuous sequence of $(i+1)dB + B$ packets leaving S_i, entering segment S_{i+1} as of time t_{i+1}. Since Starting from this time, during a period of $((i+1)+1)dB = (i+2)dB$ time units, the adversary injects one packet in every time step to the first node of S_{i+1}. It follows that during $(i+1)dB + B$ time units, starting from time t_{i+1}, there are two packets arriving to the first node of segment S_{i+1} - one on the incoming link, and one injected by the adversary. By the en-route assumption, none of the packets injected by the adversary during these $(i+1)dB + B$ time units are forwarded from this node until the entire sequence of packets arriving on the incoming link has ended. Since $(i+1)dB + B > B$, the greedy policy can store only B of the packets injected to the first node of S_{i+1} during these $(i+1)dB + B$ time steps, and must drop the remaining packets. In the following $dB - B$ time steps, there is only one packet arriving to the first node of segment S_{i+1} - the packet injected by the adversary. The first node of segment S_{i+1} therefore forwards one packet in every time step during these $dB - B$ time units as well, and maintains a full buffer during all this time. In the following B time units, the first node of segment S_{i+1} flushes its buffer, and forwards one packet in every time step. The packets forwarded by the first node of segment S_{i+1} therefore form a continuous sequence of $(i+1)dB + B + (dB - B) + B = (i+2)dB + B = ((i+1)+1)dB + B$ packets, which by the definition of the length of S_{i+1}, start arriving to the first node of S_{i+2} in time $t_{i+1} + (i+1)d = t_{i+2}$, which completes the proof of the lemma. □

Theorem 4.3.8. *For any $\frac{16B}{n} < r \leq B$ there exists an r-adversary A such that the ratio between the throughput of A and that of the greedy policy is $\Omega\left(\sqrt{\frac{rn}{B}}\right)$.*

Proof. We consider the line as divided into equal-length segments, each of a length to be determined later. Given any rate $\frac{16B}{n} < r \leq B$, we describe an adversary that inject at most one packet in every time step to every segment, and further show that the adversary does not inject to more than r segments in every time unit. This ensures that the injection sequence indeed corresponds to an r-adversary. The analysis will show that the overall number of packets accepted by the greedy policy is proportional to r times the segment length, whereas the adversary can accept all the packets injected.

Formally, Let $\frac{16B}{n} \leq r \leq B$, and let $d = \lfloor\sqrt{\frac{nB}{r}}\rfloor$. Consider the line as composed of $k = \lfloor\frac{n}{d}\rfloor = \Theta(\sqrt{\frac{rn}{B}})$ segments $S_0, \ldots, S_{k-1}$, each of length d. Note that by our assumption that $\frac{16B}{n} < r$, we are guaranteed to have $d < \frac{n}{4}$, and $k \geq 4$. We describe the sequence of injections generated by an r-adversary A:

For every $i = 0, \ldots, k-1$, A injects $\lfloor rd \rfloor$ packets to the first node of segment S_i, starting at time id.

Note that the above adversary injects at most one packet into every segment in every time step, and does not inject into more than r segments simultaneously. It follows that the above injection sequence corresponds to an r-adversary. The following lemma, whose proof appears in the sequel below, enables us to bound the number of packets leaving every segment S_i under the greedy policy.

Lemma 4.3.9. *Under the greedy policy, the packets leaving segment S_i form a continuous sequence of $\lfloor rd \rfloor + iB$ packets, which start arriving to S_{i+1} in time $d(i+1)$.*

It follows that the greedy policy absorbs and delivers $O(rd + kB) = O(\sqrt{rnB})$ packets.

The number of packets injected to the network by the adversary is $\lfloor rd \rfloor \cdot \lfloor\frac{n}{d}\rfloor = \Theta(rn)$, and the adversary may successfully deliver them all by storing the $\lfloor rd \rfloor$ packets injected to segment S_i in the buffers of that segment, and not forwarding any packet across different segments until the injection sequence has terminated.

It follows that the ratio between the number of packets delivered by the adversary and the number of packets absorbed by the greedy policy is $\Omega\left(\frac{rn}{\sqrt{rnB}}\right) = \Omega\left(\sqrt{\frac{rn}{B}}\right)$. This completes the proof of the theorem. □

Proof of Lemma 4.3.9. The proof is by induction on i. For the base case, clearly the sequence of packets leaving S_0 forms a continuous sequence of $\lfloor rd \rfloor = \lfloor rd \rfloor + 0 \cdot B$ packets, and by the choice of the length of every segment, it starts arriving to the first node of segment S_1 in time $d = (0+1)d$.

For the inductive step, assume the claim holds for i. It follows that there is a continuous sequence of $\lfloor rd \rfloor + iB$ packets leaving S_i, entering segment S_{i+1} as of time $d(i+1)$. Starting from this time, during a period of $\lfloor rd \rfloor$ time units, the adversary injects $\lfloor rd \rfloor$ packets to the first node of S_{i+1}. Since during all this time, by the induction hypothesis, there are packets arriving to the first node of S_{i+1} from its preceding node, then due to the en-route assumption, none of these packets are forwarded from this node until the entire sequence of packets arriving on the incoming link has ended. Note that since $\frac{16B}{n} < r \leq B$, we are guaranteed to have $rd > 3B$, which in turn implies that $\lfloor rd \rfloor \geq 3B > B$. [1] This implies that the greedy policy cannot store all the packets injected to the first node of segment S_{i+1}. Since the node can only store B out of the $\lfloor rd \rfloor$ packets injected by the adversary, these packets 'join' the sequence arriving on the incoming link, thus the continuous sequence of packets leaving the node comprises of $\lfloor rd \rfloor + iB + B = \lfloor rd \rfloor + (i+1)B$ packets. By the fact that the length of S_{i+1} is d, this sequence starts entering segment S_{i+2} as of time $d(i+1) + d = d(i+2)$, which completes the proof of the lemma. □

[1] This holds since for $\frac{16B}{n} < r \leq B$, we have $rd = r\lfloor\sqrt{\frac{nB}{r}}\rfloor = r\left(\sqrt{\frac{nB}{r}} - \varepsilon\right) = r\sqrt{\frac{nB}{r}} - \varepsilon r = \sqrt{rnB} - \varepsilon r > 4B - \varepsilon r > 4B - B = 3B$, where $\varepsilon < 1$.

Aiello et al. present in [5] an $\Omega(\sqrt{n})$ lower bound on the competitive ratio of the greedy policy, which is independent of B, by presenting an adversary which can deliver all the packets it injects, while any greedy policy cannot deliver more than an $O(\sqrt{n})$ fraction of the packets injected. The following lemma shows a bound on the rate of this adversary.

Lemma 4.3.10. *For any buffer size B, there exists an adversary A with rate $r = \min\{B, \sqrt{n}\}$, such that the ratio between the throughput of A and that of the greedy policy is $\Omega(\sqrt{n})$.*

Proof. The adversary used by Aiello et al. in the proof that the greedy policy cannot have competitive ratio better than $\Omega(\sqrt{n})$, is a special case of the adversary described in Section 4.3.2. The main difference is that their adversary uses a "stretch" factor of $d = 1$, instead of the factor $c\frac{n}{r^2}$ used in Section 4.3.2.

Formally, the adversary considers the line as divided into k blocks, $S_1, \ldots, S_k$, such that the length of block S_i is i, and it injects iB packets into the first node of S_i, starting from time

$$t_i = \sum_{j=1}^{i} j = \frac{i(i+1)}{2}.$$

Note that the number of segments k must satisfy $\sum_{i=1}^{k} i \leq n$. We can therefore choose $k = \lfloor\sqrt{n}\rfloor$.

Clearly this adversary has rate at most $\sqrt{n}$, since the number of segments is at most $\sqrt{n}$, and it injects at most one packet to every segment in every time unit.

We now show that the rate of this adversary is bounded by B. Note that for every i we have

$$\begin{aligned} t_{i+B} - t_i &= \tfrac{(i+B)(i+B+1)}{2} - \tfrac{i(i+1)}{2} = \\ &= \tfrac{1}{2}\left((i+B)(i+B+1) - i(i+1)\right) = \\ &= \tfrac{1}{2}\left(i^2 + 2iB + B^2 + i + B - i^2 - i\right) = \\ &= \tfrac{1}{2}\left(B^2 + (2i+1)B\right) > iB. \end{aligned}$$

It follows that by the time there are packets injected to segment S_{i+B}, there are no longer packets injected to any segment S_j, for $j \leq i$. Hence, the adversary has rate at most B since the number of segments to which it injects simultaneously is at most B. □

4.3.3 Tight Results for High Rates

In this section we conclude the results of the previous sections and derive bounds, which are tight up to constant factors, on the competitive ratio of the greedy policy for any r-adversary such that $r > 1$. We distinguish between several ranges for r. See Table 4.1 for a summary of the results.

For the range of $r \geq \min\{B, \sqrt{n}\}$, the upper bound appearing in [11] guarantees a competitive ratio of $O(\sqrt{n})$. By Lemma 4.3.10, for this range of r, there exists an r-adversary which shows that the greedy policy cannot have a competitive ratio better than $\Omega(\sqrt{n})$.

The remaining range to consider is when $1 < r < \min\{B, \sqrt{n}\}$. Assume first that $\max\left\{1, \frac{n}{B}\right\} < r < \min\{B, \sqrt{n}\}$. Theorem 4.3.3 gives an upper bound of $O(r)$. Theorem 4.3.6 gives a lower bound of $\Omega(r)$ for the case $r > 4$ (if $r \leq 4$ the upper bound guaranteed by Theorem 4.3.3 is $O(1)$). Assume now that $1 < r \leq \frac{n}{B}$. Theorem 4.3.2 gives an upper bound of

$O\left(\sqrt{\frac{rn}{B}}\right)$. Theorem 4.3.8 gives a lower bound of $\Omega\left(\sqrt{\frac{rn}{B}}\right)$, for $r > \frac{16B}{n}$ (if $r \leq \frac{16B}{n}$ the upper bound guaranteed by Theorem 4.3.2 is $O(1)$).

4.4 The Case of $B = 1$

The case of $B = 1$ is a special case for which the competitive ratio of the greedy protocol is bad. For rates $r \geq 1$ it follows easily from Theorems 4.2 and 5.1 in [5] that the competitive ratio of the greedy protocol is $\Theta(n)$. For $r \leq 1/n$ the greedy policy is optimal, since every packet is delivered before the next one can be injected. For $1/n < r < 1$ we have the following theorem:

Lemma 4.4.1. *The greedy policy has competitive ratio $\Theta(rn)$ against any r-adversary such that $1/n < r < 1$.*

Proof. Assume $B = 1$. we describe a charging scheme, which charges every packet accepted by the adversary and dropped by the greedy policy, to at least one packet accepted by the greedy policy. We further show that every packet accepted by the greedy policy is responsible for losing at most $O(rn)$ packets accepted by the adversary and dropped by the greedy policy, thus showing that the greedy policy is $O(rn)$ competitive. The charging scheme works as follows: For any packet accepted by the adversary, and not accepted by the greedy policy, injected at time t, we increase the weight of *every* packet in the network under the greedy policy by 1. Note that any packet accepted by the greedy policy, is delivered by at most n time units after its injection time. During these n time units, the adversary can inject at most $\lceil rn \rceil$ packets. It follows that no packet will have weight greater than $\lceil rn \rceil$. It follows that the competitive ratio of the greedy policy against any r-adversary with $r < 1$ is $O(rn)$.

For the lower bound, let $r' = \frac{1}{\lceil 1/r \rceil}$. Note that $\frac{r}{2} \leq r' \leq r$, and that $\frac{1}{r'}$ is integral. Consider the following r'-adversary (which by the definition of r' is clearly also an r-adversary): For every $i = 0, \ldots, \lfloor r'n \rfloor$, at time $\frac{i}{r'}$ the adversary injects one packet to node $\frac{i}{r'}$. Under the greedy policy, the packet injected to node 0 would arrive to node $\frac{i}{r'}$ at time $\frac{i}{r'}$, hence by our en-route assumption, and the fact that $B = 1$, the greedy policy cannot accommodate the new packet injected to node $\frac{i}{r'}$. It follows that the greedy policy can only deliver the first packet injected. Notice however that the adversary can deliver all packets by not forwarding any packets until time $\frac{\lfloor r'n \rfloor}{r'}$, and then deliver all packets one by one. It follows that the ratio between the number of packets delivered by the adversary and the number of packets delivered by the greedy policy is $\Omega\left(\frac{\lfloor r'n \rfloor}{1}\right) = \Omega(rn)$. □

4.5 Discussion

In this work we are interested in the question of how does the size of the buffers deployed in the network, and the injection rate of the traffic into the network, influence the attainable throughput-competitive ratio of scheduling and admission protocols. We initiate a study in the framework of the CNT model of a more refined analysis of the competitive ratio of the throughput that takes into account not only the size of the network but also the size of the buffers and the rate of injection of the traffic. We study the special case of the line network and the problem of information gathering (all packet are destined to the same node), and give

tight bounds on the competitive ratio as a function of these parameters. Interestingly, these bounds are different for different combinations of buffer-size and adversary-rate. For example, we show that for very small rates, insufficient buffer size may be the difference between the greedy protocol achieving optimal throughput, and non-optimal throughput. Furthermore, for larger rates, we show that increasing the buffer size may help up to a certain point, whereas any further increase no longer helps the greedy protocol to achieve a better competitive ratio, and its performance depends solely on the rate of the adversary.

We believe that the questions and analysis introduced in this work may lead to a better understanding of the interplay between the buffer size and the adversary rate, and the competitive ratio attainable by local-control protocols. Our work raises several interesting open problems. For example, can similar results be obtained for more involved topologies, and other protocols. Another interesting question is whether one can design protocols that would take advantage of the given buffer size in order to reduce the competitive ratio when possible.

Chapter 5

Soft Collisions in Wireless Networks

5.1 Introduction

Wireless networks often involve the joint usage of common communication channels, in a multiple access environment. In most of the models describing such settings, simultaneous transmission by more than one station results in a collision which causes all transmissions at that time to fail. Methods like collision detection (CSMA/CD) and collision avoidance (CSMA/CA), as well as other related protocols like the various variants of the Aloha protocol, are used in such scenarios in order to deal with collision, and maximize the system's throughput. In many current wireless networks, such as mesh WiFi networks, or 802.15 clusters, simultaneous usage of the same wireless channel is possible. Consider for example the settings described in Figure 5.1, where we outline two stations, A, B and their transmission ranges. If the clients of A and B are a and b respectively, then simultaneous transmissions will cause a collision at client a, while b can receive the message from B. However, if the clients of A and B are a' and b respectively, then simultaneous transmissions will both succeed, since they do not collide at either of the receiving ends.

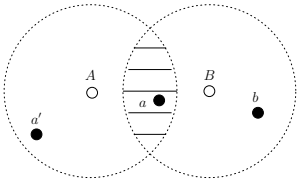


Figure 5.1: Outline of two stations, A, B and their transmissions ranges.

In this chapter we consider the problem of joint usage of a common communication channel by a finite number of stations, where stations are always backlogged, i.e., always have a packet to send. We present a generalization of classic multiple access models by introducing the notion of *soft collisions*, captured by pairwise interferences between the various stations contending for the common resource. Our contribution is twofold:

- We conduct an analytical investigation of the performance of the network from a game-theoretic point of view, where we assume stations, also called *agents*, are selfish and each agent aims at maximizing its utility. This study deals with the case where interferences

are homogeneous, i.e., any two agents inflict the same amount of interference upon each other. To evaluate the throughput of the system, we use the notions of *price of anarchy* [50] and *price of stability* [12, 26], which compare the system's performance in Nash equilibrium, (where no agent can do better by altering its chosen strategy), to the optimal throughput of the system. We show that if the utility of every agent is represented by its transmission success probability (or alternatively, its actual allocated bandwidth - see Section 5.1.1), then the performance may degrade up to an exponential factor due to the uncoordinated and selfish behavior of the agents, compared to the optimal throughput. We further present a penalty mechanism to be cast on the agents, which induces a coordinated Nash equilibrium, for which the degradation in bounded by a mere constant, thus exhibiting up to an exponential improvement compared to the uncoordinated case.

- Based on this theoretical study, we present a fully distributed protocol for heterogeneous interferences, and compare its performance with several natural protocols for the problem. Using simulations we show that our proposed protocol is superior in terms of throughput for both low and high load network conditions.

In the remainder of this section we give a formal definition of the underlying model of interferences, present a summary of our results, and discuss related work. In sections 5.2 and 5.3 we present our results for homogenous interferences, and in Section 5.4 we present our new protocol for non-homogeneous interferences, along with a simulation study of its performance, and the performance of several other protocols for non-homogenous interferences. Finally, in Section 5.5 we present some conclusions and discuss open questions.

5.1.1 Model

Assume we have n agents using a common wireless medium. For every agent i, we let $[0,1]$ be the strategy space of agent i, and let $R_i \in [0,1]$ denote a strategy chosen by agent i. We refer to R_i as the *probability* that agent i transmits. R_i can also be considered as the *rate* in which agent i transmits. Due to interferences, the probability of a *successful* transmission also depends upon the transmission of other agents, or alternatively, the *effective rate* an agent eventually experiences depends upon the transmission rates of the other agents. Given a profile $\overline{R} = (R_1, \ldots, R_n) \in [0,1]^n$, we define the *success probability* of agent i's transmission (or alternatively, the *effective rate* obtained by agent i) as:

$$r_i(\overline{R}) = R_i \cdot \Pi_{j \neq i}(1 - \alpha_{i,j} R_j),$$

where for every $1 \leq i, j \leq n$, $\alpha_{i,j} \in [0,1]$ is a fixed network-dependent parameter denoting the amount of interference inflicted on i upon simultaneous transmission of both i and j.

Clearly, if for all i, j, $\alpha_{i,j} = 0$, i.e., there are no interferences, then in any reasonable setting the selfish behavior of the agents will result in all agents transmitting at maximum rate, which implies an optimal use of resources. On the other hand, if for all i, j, $\alpha_{i,j} = 1$, then our model coincides with classic multiple access models, where in any case of simultaneous transmissions a collision occurs, resulting in the failure of all the transmissions. We refer to the interferences as *homogeneous* if there exists some $\alpha \in (0,1)$ such that for all i, j, $\alpha_{i,j} = \alpha$. If this is not the

case, then we refer to the interferences as *non-homogeneous*. By the above observations, when considering homogeneous interferences, we restrict our attention to the case where for all i, j, $\alpha_{i,j} = \alpha \in (0, 1)$. Given an agent i, we let the set of its *neighbors* consist of all agents j such that $\alpha_{i,j} > 0$.

Given a profile $\overline{R} = (R_1, \ldots, R_n)$, the *social welfare* $\varphi(\overline{R})$ is considered to be the overall use of resources in the system, i.e.

$$\varphi(\overline{R}) = \sum_{i=1}^{n} r_i(\overline{R}) = \sum_{i=1}^{n} R_i \prod_{j \neq i} (1 - \alpha_{i,j} R_j).$$

$\varphi(\overline{R})$ can be interpreted as the expected number of successful transmission, or alternatively, as the overall usage of bandwidth in the network.

For every agent i, we let $U_i(\overline{R})$ be the utility function of agent i, assuming agents play profile $\overline{R}$. In the following sections we consider several choices for these utility functions, and discuss the system's performance where agents are selfish, and aim at maximizing their own utility, regardless of the effect their choices have on the overall social welfare. We refer to the above setting as the *soft collisions multiple access (SCMA) game*.

Given any profile $\overline{R} = (R_1, \ldots, R_n)$, we let $\overline{R}_{-i}$ denote the subprofile defined by strategies of all agents save agent i. We further denote by $(\overline{R}_{-i}, R'_i)$ the profile where every agent other than i plays the same strategy as in $\overline{R}$, while agent i plays strategy R'_i. A profile $\overline{R}$ is said to be a *Nash equilibrium* if for every i, and every $R'_i \in [0, 1]$, $U_i(\overline{R}) \geq U_i(\overline{R}_{-i}, R'_i)$. Intuitively, a profile is at Nash equilibrium if no agent can increase its benefit by unilaterally deviating from his choice. Given any $n \in \mathbb{N}$, we use $\overline{R}^{(n)}_{\mathrm{NE}}$ to denote a Nash equilibrium profile for n agents, and use $\overline{R}^{(n)}_{\mathrm{OPT}}$ to denote any profile for n agents which maximizes the social welfare. Assuming a Nash equilibrium exists, we use the notion of *Price of Anarchy (PoA)* in order to evaluate this effect, defined by the supremum over all Nash equilibria $\overline{R}^{(n)}_{\mathrm{NE}}$ of the ratio between $\varphi(\overline{R}^{(n)}_{\mathrm{OPT}})$ and $\varphi(\overline{R}^{(n)}_{\mathrm{NE}})$, capturing the performance of the worst case equilibrium. We further consider the notion of *Price of Stability (PoS)*, defined by the infimum of the above ratio over all Nash equilibria, capturing the performance of the best case equilibrium.

5.1.2 Our Results

We study the rational choices of the agents in an SCMA setting, and analyze the performance of Nash equilibria compared to the optimal performance. For homogeneous interferences, we show that when the utility of an agent is its effective rate, then selfishness may cause the system's performance to be up to an exponential factor far from the optimal performance. Specifically, we show that for any constant α, the price of anarchy as well as the price of stability are exponential in the number of agents, i.e., *any* equilibrium suffers an exponential degradation in performance. These results appear in Section 5.2.

We then show that there exists a penalty function which is proportional to the amount of aggressiveness demonstrated by an agent, such that for the case where the utility of an agent is the sum of its rate and its penalty, then the price of stability with regards to the coordinated equilibria can be made to drop to at most $e \approx 2.718$, thus demonstrating that an exponential

improvement is possible compared to the uncoordinated case. We further show that for interferences which are not too large, namely, for $\alpha \leq 2/e$, the price of anarchy is also bounded by e, thus ensuring that the degradation in performance due to the selfishness of the agents can be guaranteed to be made very small. These results mean that if we impose these penalties upon the agents, either in the form of payment for transmission to the network operator, or considering them as an intrinsic cost suffered by the agent due to transmission (e.g., due to power consumption), then the performance can be improved dramatically compared to the general case where the agent's utility is merely its effective rate. These results are presented in Section 5.3.

For the general version of SCMA, where interferences are non-homogeneous, we devise a new protocol, and examine its performance via extensive simulations. Our new protocol, which is motivated by our analytic analysis of the homogeneous setting, is shown to obtain higher throughput than all other protocols for the problem. Moreover, our protocol is very simple, fully distributed, and very robust to load. That is, it performs well both under low load and high load conditions. These results show that our new protocol obtains high throughput regardless of the number of agents. This also renders it the best choice for networks with a varying number of agents. This study is presented in Section 5.4.

5.1.3 Previous Work

There have been several studies of the selfish behavior of agents in multiple access environments. The slotted Aloha model was studied in, e.g., [7, 9, 37, 53–55, 58], in the Markovian setting, presenting conditions for the system's stability, and investigating convergence to an equilibrium. Some recent work [57, 71] has also considered the role of introducing costs for transmissions, and its effect on the stability of the system. Other aspects of selfish behavior in CSMA/CA networks, such as the effect of agents' selfish deviation from the protocol, were studied in [69].

There has been much interest in game theoretic perspectives on combinatorial optimization problems, most notably following the introduction of the notion of price of anarchy in [50]. Recent works have considered the role of taxation, or penalties, on the performance of non-cooperative systems with selfish agents, thus resulting in *coordinated Nash equilibria*. For the problem of selfish routing, first discussed in [66], where the objective is to have the minimum overall latency of a given set of connections, it was known that tolls may contribute to obtaining a smaller overall latency, in a setting where each selfish agent aims at minimizing his utility, comprising of its latency together with an appropriate toll [20]. In consecutive studies it was shown that such tolls may be found in an efficient manner, both in terms of verifying their existence, as well as maintaining them to the lowest level possible [24, 30]. Additional work using the notion of coordinated equilibria appears in [23] where they further introduce coordination mechanism for the problem of load balancing a set of jobs on machines with the aim of minimizing the makespan, which results in a substantial improvement compared to the uncoordinated settings.

Another type of equilibria in the framework of the Aloha model is discussed in [8], where they consider the notion of *correlated Nash equilibria*, in which an arbitrator may send some random signal to every agent, that can be used by the agent in choosing its strategy, such that the

overall system's performance improves by the agent's using the additional information provided by the arbitrator, thus introducing some coordination.

Another recent study focused on the time it takes all the agents to successfully transmit one packet each [29]. They present a protocol which is in equilibria and uses "self incentive" type of penalties, while succeeding at delivering all packets w.h.p. in linear time.

Our model seems especially applicable to static wireless networks, such as vastly studied wireless mesh networks, and to wireless personal area networks (WPAN) based on the new IEEE 802.15.4 (Zigbee) standard. In the former case, there has been much interest recently in distributed protocols which enable maximizing the throughput by avoiding collisions [22, 59]. In the latter, not only does CSMA/CA constitute a substantial part of the MAC layer, but the system's performance also depends heavily upon maintaining synchronization in the network using beacon frames. The behavior of the proposed synchronization scheme is in itself a multiple access environment. The standard does not fully specify the protocols to be used in this respect [49]. We believe that both our model, and our protocol suggested in Section 5.4 can be used to address these challenges.

5.2 Homogeneous Interferences - General Nash Equilibria

In this section we present several analytic results as to the effect of selfishness upon the performance of the network, for the case where interferences are homogeneous, i.e., for every i, j, $\alpha_{i,j} = \alpha$, for some system's parameter $\alpha \in (0, 1)$. We first consider the simple utility function $U_i(\overline{R}) = r_i(\overline{R})$, and show that in such a case, the system's performance can be very far from optimal. The following lemma is straight forward and follows immediately from the definition of the utility function:

Lemma 5.2.1. *For utility functions $U_i(\overline{R}) = r_i(\overline{R})$, the only Nash equilibrium solution is obtained where every agent i plays the strategy $R_i = 1$. The social welfare value of this Nash equilibria is $n(1-\alpha)^{n-1}$.*

Lemma 5.2.1 implies in particular that since there is only one Nash equilibrium solution in these settings, then the price of stability equals the price of anarchy. In order to determine this value, we now turn to analyze the value of a profile which maximizes the social welfare. We first analyze the value of the social welfare function on the boundary of the profiles domain $[0, 1]^n$, and then turn to analyze the maximum value obtained in the interior of the domain.

Since φ is symmetric, any two integral profiles $\overline{R}, \overline{R}' \in \{0, 1\}^n$ which have the same number of 1's, satisfy $\varphi(\overline{R}) = \varphi(\overline{R}')$. Let $B_k = (R_1, \ldots, R_n)$ denote any profile with exactly k 1's. It therefore follows that the value in every extreme point where k agents play the 1-strategy and $n - k$ agents play the 0-strategy, is given by

$$\begin{aligned} v_k = \varphi(B_k) &= \textstyle\sum_{i:R_i=1} R_i \prod_{j\neq i}(1-\alpha R_j) \\ &\quad + \textstyle\sum_{i:R_i=0} R_i \prod_{j\neq i}(1-\alpha R_j) \\ &= \textstyle\sum_{i:R_i=1} \prod_{j\neq i, R_j=1}(1-\alpha) \\ &= k(1-\alpha)^{k-1}. \end{aligned}$$

The following lemma shows an ordering of the values of v_k, depending on the value of α.

Lemma 5.2.2. *$v_k > v_{k-1}$ if and only if $\alpha < \frac{1}{k}$.*

Proof. Requiring that $v_k > v_{k-1}$ is equivalent to requiring $v_k/v_{k-1} > 1$. By the definition of v_k, this holds if and only if $\frac{k(1-\alpha)^{k-1}}{(k-1)(1-\alpha)^{k-2}} > 1$, which by further simplification results in $1-\alpha > \frac{k-1}{k}$. This indeed holds if and only if $\alpha < \frac{1}{k}$. □

The following is an immediate corollary of Lemma 5.2.2:

Corollary 5.2.3. *If $\alpha \in [\frac{1}{k+1}, \frac{1}{k})$ then $\max_j v_j = v_k$.*

Figure 5.2 outlines the different values of v_k, as a function of α. Recall that v_k is the social value when exactly k agents transmit, and all other $n-k$ agents refrain from transmitting.

Since clearly $\varphi(\overline{R}^{(n)}_{\mathrm{OPT}}) \geq v_k$ for all k and for all α, we therefore have $\varphi(\overline{R}^{(n)}_{\mathrm{OPT}}) \geq \max_k v_k$ for all α. In order to show that indeed $\varphi(\overline{R}^{(n)}_{\mathrm{OPT}}) = \max_k v_k$, we wish to show that the maximum of $\varphi(\cdot)$ is not obtained in the interior of the domain. We first show that there is only one possible extreme point x_0 in the interior of the domain $(0,1)^n$. We further show that $\varphi(x_0) \leq \max_k v_k$, which therefore implies that $\varphi(\overline{R}^{(n)}_{\mathrm{OPT}}) = \max_k v_k$ for all α.

Lemma 5.2.4. *In the n-agents SCMA game, given any $\alpha \in (0,1)$, the only possible extreme point of the social welfare function $\varphi(\cdot)$ in the interior of $(0,1)^n$ is $\overline{R}_0 = (\frac{1}{\alpha n}, \ldots, \frac{1}{\alpha n})$.*

Proof. Note that by algebraic simplification we have

$$\begin{aligned}
\varphi(\overline{R}) &= \textstyle\sum_{i=1}^{n} R_i \prod_{j\neq i}(1-\alpha R_j) \\
&= R_n \textstyle\prod_{j\neq n}(1-\alpha R_j) \\
&\quad + \textstyle\sum_{i\neq n} R_i \prod_{j\neq i}(1-\alpha R_j) \\
&= R_n \textstyle\prod_{j\neq n}(1-\alpha R_j) \\
&\quad + (1-\alpha R_n) \textstyle\sum_{i\neq n} R_i \prod_{j\neq i,n}(1-\alpha R_j) \\
&= R_n \Big(\textstyle\prod_{j\neq n}(1-\alpha R_j) \\
&\quad - \alpha \textstyle\sum_{i\neq n} R_i \prod_{j\neq i,n}(1-\alpha R_j) \Big) \\
&\quad + \textstyle\sum_{i\neq n} R_i \prod_{j\neq i,n}(1-\alpha R_j).
\end{aligned}$$

By taking derivatives we obtain

$$\frac{\partial \varphi}{\partial R_n} = \prod_{j\neq n}(1-\alpha R_j) - \alpha \sum_{i\neq n} R_i \prod_{j\neq i,n}(1-\alpha R_j),$$

where verifying when the above is zero and reordering we obtain (since $\alpha \in (0,1)$)

$$0 = 1 - \alpha \sum_{i\neq n} \frac{R_i}{1-\alpha R_i},$$

which in turn yields

$$\frac{\alpha R_1}{1-\alpha R_1} = 1 - \alpha \sum_{i\neq 1,n} \frac{R_i}{1-\alpha R_i}.$$

Note that the term on the right hand side is independent of R_1 and R_n. Following the same above procedure of taking first the derivative with regards to R_1, and then isolating R_n results in

$$\frac{\alpha R_n}{1-\alpha R_n} = 1 - \alpha \sum_{i \neq 1,n} \frac{R_i}{1-\alpha R_i} = \frac{\alpha R_1}{1-\alpha R_1}.$$

It therefore follows that $R_1 = R_n$. Since the above argument holds for any $1 \leq i < j \leq n$, we have that the only point where all derivatives are zero satisfies $R_i = R_j$ for all i, j.

We can therefore consider the possible extreme points of φ in the interior of the domain as a subset of the n-tuples of the extreme points of the single variable function ψ, where $\psi(x) = nx(1-\alpha x)^{n-1}$.

Taking derivatives we obtain

$$\begin{aligned} \psi'(x) &= n(1-\alpha x)^{n-1} \\ &\quad + nx(n-1)(1-\alpha x)^{n-2}(-\alpha) \\ &= [(1-\alpha x) - \alpha(n-1)x]\, n(1-\alpha x)^{n-2} = 0. \end{aligned}$$

There are two options to consider. First consider the case where $(1-\alpha x) = 0$ which implies $x = \frac{1}{\alpha}$. This is impossible in the interior of the domain since $\alpha < 1$. It therefore follows that

$$(1-\alpha x) - \alpha(n-1)x = 0.$$

It therefore follows that $x_0 = \frac{1}{\alpha n}$, which in turn implies that the only possible extreme point of $\varphi(\cdot)$ in the interior of $(0,1)^n$ is $\overline{R}_0 = (\frac{1}{\alpha n}, \ldots, \frac{1}{\alpha n})$. □

The following lemma suffices in order to show that $\varphi(\overline{R}_0) \leq \max_j v_j$.

Lemma 5.2.5. *Consider the profile $\overline{R}_0 = (\frac{1}{\alpha n}, \ldots, \frac{1}{\alpha n})$. For any $k = 1, \ldots, n-1$, if $\alpha \in [\frac{1}{k+1}, \frac{1}{k})$, then $\varphi(\overline{R}_0) \leq v_k$.*

Proof. By definition, we need to show

$$\varphi(\overline{R}_0) = \frac{(n-1)^{n-1}}{\alpha \cdot n^{n-1}} \leq k(1-\alpha)^{k-1} = v_k,$$

or equivalently, we need to show that for any $\alpha \in [\frac{1}{k+1}, \frac{1}{k})$

$$f_k(\alpha) = k \cdot n^{n-1} \cdot \alpha(1-\alpha)^{k-1} - (n-1)^{n-1} \geq 0.$$

Taking derivatives, we obtain

$$\begin{aligned} f_k'(\alpha) &= k \cdot n^{n-1}(1-\alpha)^{k-1} \\ &\quad - k \cdot n^{n-1} \cdot \alpha(k-1)(1-\alpha)^{k-2} \\ &= k \cdot n^{n-1}(1-\alpha)^{k-2}\,[(1-\alpha) - (k-1)\alpha] \\ &= k \cdot n^{n-1}(1-\alpha)^{k-2}\,[1 - k \cdot \alpha]\,. \end{aligned}$$

The derivative is zero in one of two cases:

1. $\alpha = 1$: This is out of bounds.

2. $\alpha = \frac{1}{k}$.

It follows that for $\alpha < \frac{1}{k}$, $f_k(\alpha)$ is strictly monotone increasing. It therefore suffices to show that $f_k(\frac{1}{k+1}) \geq 0$, i.e.:

$$\begin{aligned} f_k(\tfrac{1}{k+1}) &= k \cdot n^{n-1} \cdot \tfrac{1}{k+1} \cdot (1 - \tfrac{1}{k+1})^{k-1} \\ &\quad -(n-1)^{n-1} \\ &= n^{n-1} \cdot \tfrac{k}{k+1} \cdot (1 - \tfrac{1}{k+1})^{k-1} \\ &\quad -(n-1)^{n-1} \\ &= n^{n-1} \cdot \tfrac{k}{k+1} \cdot (\tfrac{k}{k+1})^{k-1} \\ &\quad -(n-1)^{n-1} \\ &= n^{n-1}(\tfrac{k}{k+1})^{k} - (n-1)^{n-1} \\ &= n^{n-1}(1 - \tfrac{1}{k+1})^{k} - (n-1)^{n-1} \geq 0, \end{aligned}$$

which follows since,

$$(1 - \frac{1}{k+1})^k \geq (1 - \frac{1}{n})^{n-1},$$

for every $k = 1, \ldots n-1$, thus completing the proof.[1] □

The above lemma combined with Corollary 5.2.3 immediately implies the following corollary:

Corollary 5.2.6. *Given n agents, if $\alpha \in [\frac{1}{k+1}, \frac{1}{k})$, then $\varphi(\overline{R}^{(n)}_{\mathrm{OPT}}) = v_k$.*

Figure 5.2 depicts the various plots of v_k as a function of α, up to $k = 5$. The maximal overall throughput as a function of α is given by the maximal curve among all v_ks. One can see that the overall throughput, which is also the optimal social welfare, increases as α approaches 0.

Combining Lemma 5.2.1 which shows that there exists a single Nash equilibrium solution $\overline{R}^{(n)}_{\mathrm{NE}} = (1, \ldots, 1)$ whose value is $\varphi(\overline{R}^{(n)}_{\mathrm{NE}}) = n(1-\alpha)^{n-1}$, along with Corollary 5.2.6, we can conclude the following theorem:

Theorem 5.2.7. *Given n agents, and any $k \in \{1, \ldots, n-1\}$,*

1. *If $\alpha \in \left[\frac{1}{k+1}, \frac{1}{k}\right)$ then*

$$\mathrm{PoA}^{(n)} = \mathrm{PoS}^{(n)} = \frac{k}{n(1-\alpha)^{n-k}}.$$

2. *If $\alpha \leq \frac{1}{n}$ then $\mathrm{PoA}^{(n)}_{\alpha} = 1$.*

Note that Theorem 5.2.7 implies that for any constant $m \in \mathbb{N}$, for $\alpha = 1/m$ we have

$$\mathrm{PoA}^{(n)} = \mathrm{PoS}^{(n)} = 2^{\Omega(n)}.$$

[1] This is due to the fact that the function $g(x) = (1 - 1/x)^{x-1}$ is monotone decreasing.

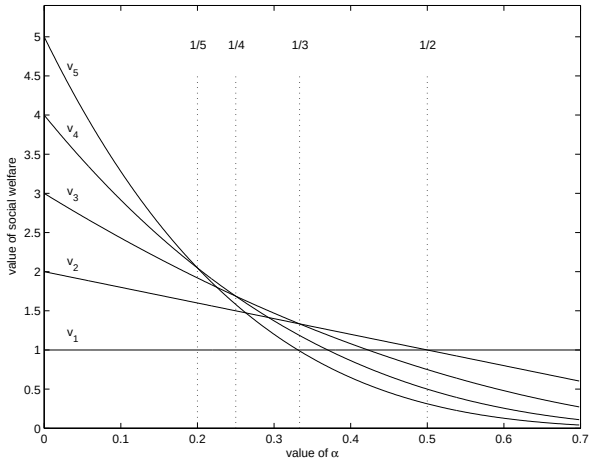


Figure 5.2: Different values of v_k, depending on the value of α, up to $k = 5$.

5.3 Homogeneous Interferences - Coordinated Nash Equilibria

In this section we introduce a penalty based scheme, where every agent i incurs a penalty $p_i(\cdot)$ for transmission. We consider two types of penalties. The first type depends upon the choices of all the agents in the system, i.e., $p_i(\overline{R})$, while the second type only depends upon the choice of agent i, i.e., $p_i(R_i)$. The general form of the utility function of agent i is therefore $U_i(\overline{R}) = r_i - p_i$. We use the notion of coordinated Nash equilibrium and show that for both penalty functions, the selfishness of the agents does not result in more than a constant factor degradation in performance compared to the optimal performance. This should be contrasted with the results presented in the previous section showing that the price of stability for the uncoordinated case can be exponential in the number of agents.

We first show that if for every $q \in [0, 1]$, we can coerce all players to play strategy q, then there exists a choice for q such that the degradation in performance is bounded by a constant. We then show that there exist penalty functions which cause such a profile to be at Nash equilibrium. It follows that selfishness can be tamed into providing a throughput that is at most a constant factor far from the optimal throughput.

5.3.1 Can one strategy suit all?

Given any $q \in [0, 1]$, let $\overline{R}^q$ denote a profile where all agents play strategy q. Given some value q, assume that we are able to choose some penalty, such that $\overline{R}^q$ is at Nash equilibrium.

Note that the social welfare value of any such profile $\overline{R}^q$ is given by the function

$$\psi(q) = nq(1 - \alpha q)^{n-1}.$$

As shown in the proof of Lemma 5.2.4, the value of q which maximizes $\psi(\cdot)$ is $q_0 = \frac{1}{\alpha n}$. It follows that the social welfare value of the profile $\overline{R}^{q_0}$ is

$$\psi(q_0) = \frac{1}{\alpha}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{e\alpha},$$

where the inequality follows from the fact that $\left(1-\frac{1}{n}\right)^{n-1}$ is strictly monotone decreasing, and converges to e^{-1}.

As we have seen, the optimal value of the social welfare function for $\alpha \in [\frac{1}{k+1}, \frac{1}{k})$ [2] is obtained for a profile where k agents play the 1-strategy, and $n-k$ agents play the 0-strategy, resulting in a social welfare value of

$$\varphi(\overline{R}^{(n)}_{\mathrm{OPT}}) = k(1-\alpha)^{k-1} < k\left(1-\frac{1}{k}\right)^{k-1} \leq \frac{1}{\alpha}.$$

It follows that

$$\frac{\varphi(\overline{R}^{(n)}_{\mathrm{OPT}})}{\varphi(\overline{R}^{q_0})} \leq e.$$ [3]

In the following section we indeed show that we can choose a penalty function such that the profile $\overline{R}^{q_0}$ is at Nash equilibrium.

5.3.2 Things are simple if penalties depend on others

Let $q \in [0,1]$, and consider the following utility function:

$$U^q_i(\overline{R}) = R_i \prod_{j \neq i}(1-\alpha R_j)(2q - R_i),$$

which can be cast as a utility function of the form

$$U^q_i(\overline{R}) = r_i(\overline{R}) - p^q_i(\overline{R}),$$

where $p^q_i(\overline{R}) = \prod_{j\neq i}(1-\alpha R_j)((1-2q)R_i + R_i^2)$. Assume all agents except for i play strategy q. It follows that

$$\begin{aligned} U^q_i(R) &= R_i(1-\alpha q)^{n-1}(2q - R_i) \\ &= (1-\alpha q)^{n-1}(2qR_i - R_i^2). \end{aligned}$$

By taking derivatives, we obtain that the maximum is obtained for $R_i = q$, i.e., $\overline{R}^q$ is at Nash equilibrium.

It therefore follows that the price of stability is at least

$$\max_q \frac{\varphi(\overline{R}^{(n)}_{\mathrm{OPT}})}{\varphi(\overline{R}^q)}.$$

In addition, since choosing $R_i = q$ is the best response of agent i regardless of the strategy chosen by any agent $j \neq i$, we can conclude that $\overline{R}^q$ is the only Nash equilibrium solution, hence the price of anarchy is the same as the price of stability.

Combining this result with the result presented in the previous section, for $q = \frac{1}{\alpha n}$, we obtain the following theorem:

Theorem 5.3.1. *For every agent i there exists a penalty function $p_i(\overline{R})$ for which the price of anarchy' as well as the price of stability, are at most e.*

[2] Equivalently, $k < \frac{1}{\alpha} \leq k+1$.

[3] Note that for α sufficiently distant from 1, a much better bound can be obtained. E.g., already for $\alpha < \frac{1}{2}$ this analysis yields that the ratio is at most $e/2$.

5.3.3 Only one's strategy affects the penalty

In this section we use the intuition underlying the proof of Theorem 5.3.1 in order to present for every agent i a penalty function p_i which is independent of the strategy adopted by any agent other than i, i.e., the penalty function of agent i depends only on R_i.

Specifically, given any $q \in [0,1]$, we consider for every agent i the utility function

$$U_i^q(\overline{R}) = r_i(\overline{R}) - p_i^q(R_i),$$

where

$$p_i^q(R_i) = (1-\alpha q)^{n-1}((1-2q)R_i + R_i^2).$$

Assume all agents but i play the strategy q. It follows that the utility function of agent i is the same as in the previous section, i.e.,

$$\begin{aligned} U_i^q(R_i) &= R_i(1-\alpha q)^{n-1}(2q - R_i) \\ &= (1-\alpha q)^{n-1}(2qR_i - R_i^2), \end{aligned}$$

which in turn implies that the best strategy for agent i to play is $R_i = q$, hence $\overline{R}^q$ is at Nash equilibrium. Similarly to Theorem 5.3.1, we thus obtain the following theorem:

Theorem 5.3.2. *For every agent i there exists a penalty function $p_i(R_i)$ for which the overall price of stability is at most e.*

In what follows we analyze the conditions for which $\overline{R}^q$ is actually the *only* Nash equilibrium. In any case for which there is a single Nash equilibrium, the price of stability is the same as the price of anarchy, which provides a guarantee upon the worst case equilibrium. Specifically, we prove the following theorem:

Theorem 5.3.3. *If $\alpha \leq \frac{2}{e}$, then for every agent i there exists a penalty function $p_i(R_i)$ for which the overall price of anarchy is at most e.*

Proof. By considering the derivative of the utility function of agent i, we obtain that

$$\frac{\partial U_i^q}{\partial R_i} = \prod_{j \neq i}(1-\alpha R_j) - (1-\alpha q)^{n-1}(1-2q+2R_i) = 0,$$

which implies

$$R_i = \frac{\prod_{j\neq i}(1-\alpha R_j)}{2(1-\alpha q)^{n-1}} - (\frac{1}{2} - q).$$

By further isolating R_k for some $k \neq i$, we obtain

$$\begin{aligned} R_i &= (1-\alpha R_k)\tfrac{\prod_{j\neq i,k}(1-\alpha R_j)}{2(1-\alpha q)^{n-1}} - (\tfrac{1}{2} - q) \\ &= \underbrace{\left[\frac{\prod_{j\neq i,k}(1-\alpha R_j)}{2(1-\alpha q)^{n-1}} - (\frac{1}{2} - q)\right]}_{a_{i,k}} \\ &\quad - \underbrace{\left[\alpha \cdot \frac{\prod_{j\neq i,k}(1-\alpha R_j)}{2(1-\alpha q)^{n-1}}\right]}_{b_{i,k}} R_k. \end{aligned}$$

Note that $a_{i,k}$ and $b_{i,k}$ are independent of R_i and R_k, hence we can write the relation between these two as

$$R_i = a_{i,k} - b_{i,k}R_k.$$

By applying the same arguments, starting with considering the derivative $\frac{\partial U_k^q}{\partial R_k}$, and then isolating R_i, we can obtain that in the point where both derivatives are zero we must also have

$$R_k = a_{i,k} - b_{i,k}R_i.$$

Solving this system of linear equations we obtain that any point in which both the derivative of $U_i^q(R_i)$ and of $U_k^q(R_k)$ is zero, assuming $b_{i,k} \neq 1$, we must have $R_i = R_k$.[4] Since this applies to any i, k, we can conclude that if for all i, k, $b_{i,k} \neq 1$, then we have $R_1 = R_2 = \ldots = R_n$.

Assume now that for some $i \neq k$, $b_{i,k} = 1$. In this case, we have

$$b_{i,k} = \alpha \cdot \frac{\prod_{j \neq i,k}(1 - \alpha R_j)}{2(1 - \alpha q)^{n-1}} = 1,$$

which in turn implies

$$\prod_{j \neq i,k}(1 - \alpha R_j) = \frac{2(1 - \alpha q)^{n-1}}{\alpha}.$$

By taking $q = \frac{1}{\alpha n}$, we can conclude that

$$\prod_{j \neq i,k}(1 - \alpha R_j) = \frac{2(1 - \frac{1}{n})^{n-1}}{\alpha} > \frac{2e^{-1}}{\alpha}.$$

Since $\prod_{j \neq i,k}(1 - \alpha R_j) \leq 1$, we thus have $\alpha > \frac{2}{e}$.

It follows that for $q = \frac{1}{\alpha n}$, if $\alpha \leq \frac{2}{e} \sim 0.736$, then for all $i \neq k$, $b_{i,k} \neq 1$. In such a case, we are guaranteed to have $R_i = R_k$ for all i, k.

Assume $\alpha \leq \frac{2}{e}$, (and $q = \frac{1}{\alpha n}$) which implies that any Nash equilibrium solution must satisfy $R_i = R_k$ for all i, k. It follows that we seek a Nash equilibrium solution where the utility function of every agent i is given by

$$\tilde{U}_i^q(x) = x(1 - \alpha y)^{n-1} - x(1 - \alpha q)^{n-1}(1 - 2q + x).$$

where all other agents except for i play the same strategy y. In order for this to be in equilibrium, the following conditions must hold:

1. $(\tilde{U}_i^q)'(x) = 0$.
2. $x = y$, since in any equilibrium all agents play the same strategy.

In order to see condition (1), consider

$$(\tilde{U}_i^q)'(x) = (1 - \alpha y)^{n-1} - (1 - \alpha q)^{n-1}(1 - 2q + 2x) = 0.$$

[4] If we let $a = a_{i,k}$ and $b = b_{i,k}$, by substitution we have $R_i = a - b(a - bR_i) = a - ab + b^2 R_i = a(1-b) + b^2 R_i$ which in turn implies $(1 - b^2)R_i = a(1 - b)$, i.e., $(1 - b)(1 + b)R_i = a(1 - b)$, which implies $R_i = \frac{a}{1+b}$, assuming $b \neq 1$. Note that $b \geq 0$.

Since by condition (2) we must have $x = y$, it follows that we wish to find the solutions to the equation

$$\underbrace{(1-\alpha y)^{n-1}}_{p(y)} - \underbrace{(1-\alpha q)^{n-1}(1-2q+2y)}_{\ell(y)} = 0.$$

Clearly, $y = q$ is a solution to this equation. We now show that this is the only solution to this equation. Since $p(y)$ is strictly decreasing as a function of y, while $\ell(y)$ is strictly increasing as a function of y, any deviation from $y = q$ renders this equation false. It follows that $y = q$ is the only solution to this equation, hence the only Nash equilibrium solution is obtained for the profile $\overline{R} = (\frac{1}{\alpha n}, \ldots, \frac{1}{\alpha n})$. □

5.4 Protocols for Non-homogeneous Interferences

In this section we study the more general SCMA problem where interferences are non-homogeneous, i.e., for every two agents i, j we have an interference parameter $\alpha_{i,j}$, reflecting the interference of j with respect to i. Going back to the mesh wireless networks scenario, a transmission of access point A will fail due to a simultaneous transmission of access point B, if A's message is sent to a user that is also affected by B's transmission. If the area covered by an access point is a planar disc, and users are distributed uniformly at random in this area, then a transmission of A to a client a will fail due to B's transmission with probability proportional to the area of the intersection of the discs (the lined area in Figure 5.1), divided by the area of the disc of access point A. If all access points have identical transmission range, then the interferences are symmetric, i.e., for every i, j, $\alpha_{i,j} = \alpha_{j,i}$.

5.4.1 Protocols Description

Our new protocol INTERFERENCESRAND is motivated by the results appearing in the previous sections; if the interferences were homogeneous, then by Theorem 5.3.2 the profile where every agent chooses the strategy $\frac{1}{\alpha n} = \frac{1}{\sum_{j=1}^{n} \alpha}$ can be made into a Nash equilibrium solution by using an appropriate penalty function, and the social welfare value of such a Nash equilibrium solution can be guaranteed to cover at least a $1/e$ fraction of the optimal social welfare value. Protocol 5.1 generalizes this observation to the general (non-homogeneous) settings.

Protocol 5.1 INTERFERENCESRAND(agent i)

1: **for** all agents j that are neighbors of i **do**
2: calculate $\alpha_{i,j}$
3: **end for**
4: set $\tau_i = \frac{1}{1+\sum_{j \neq i} \alpha_{i,j}}$
5: at each slot, transmit with probability τ_i

Next we evaluate the performance of our new protocol using simulations. To do that, we compare its performance with the performance of several broadly used protocols for multiple access environments. Specifically, we consider the following additional protocols:

1. CLUSTERIZE: This is a clustering protocol, whose variant is used, e.g., in IEEE 802.15.4 (Zigbee). Agents are divided into clusters, and in every cluster, a TDM discipline is used to determine the transmission schedule. The division into clusters is done by greedily assigning agents to clusters. A yet-unassigned agent i is chosen at random, and its cluster is defined by all yet-unassigned agents j such that $\alpha_{i,j} > 0$, which are then assigned to i's cluster. This generates a *local* clustering scheme, and note that it is possible for two agents i, j which correspond to different clusters to have $\alpha_{i,j} > 0$. In each cluster agents transmit one after the other, in a Round-Robin fashion.

2. SQRTRAND: Every agent transmits with a probability proportional to the square root of the number of its neighbors. Formally, each agent i transmits with probability $\tau_i = \frac{1}{\sqrt{|\{j|\alpha_{i,j}>0\}|}}$.

3. INTERSECTRAND: Every agent transmits with a probability proportional to the number of its neighbors. Formally, each agent i transmits with probability $\tau_i = \frac{1}{|\{j|\alpha_{i,j}>0\}|}$.

4. GREEDY: All agents transmit simultaneously. As shown in Section 5.2, this protocol captures the only uncoordinated Nash equilibrium for the SCMA problem.

5. HALFRAND: Each agent i transmits with probability $1/2$.

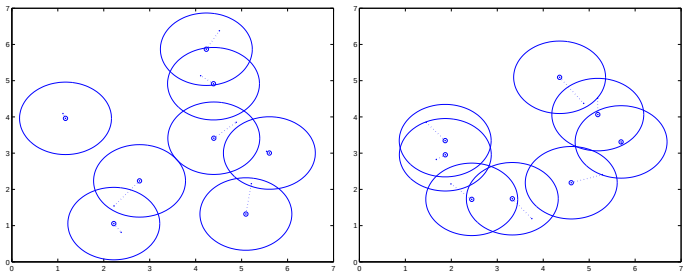
5.4.2 Simulation Description

We study the problem by considering each agent as defined by a base station positioned in the 2-dimensional plane, and let its transmission area be the unit disc centered at the base station's position. As mentioned before, this implies symmetric interferences.

Our simulation study of the performance of the above protocols consists of several components. We randomly choose a configuration of agents, by randomly and uniformly placing base stations in a bounded 2-dimensional plane. The transmission range of every base station is a unit disc centered at the base station's location. Such a configuration fully defines the interferences matrix between all agents. For every agent i defined by its unit disc location, we randomly and uniformly choose a location in its transmission range as the transmission target of the agent. Figure 5.3 shows the outline of 2 different configurations with $n = 8$ agents, each with a transmission target in its range. Every agent then decides whether or not to transmit to its target, according to the protocol used. A transmission of agent i to its target is considered successful if and only if agent i indeed transmits, and every other agent j such that i's target is in j's range, does not transmit. Reconsidering Figure 5.3, if a transmission target lays in the intersection the transmission ranges of its base station and another base station, then simultaneous transmission by both of these stations will cause the transmission to fail at the target.

We measure the *throughput* of a specific protocol for a specific choice of targets by the various agents, as the ratio between the number of successful transmissions, and the overall number of agents.

Our simulations consist of checking 10 different configurations for values of n - the number of agents - ranging from 25 to 1500, over a 40×40 domain. For each configuration we conducted 20 rounds of choosing a transmission target, and verifying the successful transmissions.



(a) (b)

Figure 5.3: Examples of different agents configurations, each with a transmission target ($n = 8$).

5.4.3 Simulation Results

Figure 5.4 shows the average throughput results for protocol INTERFERENCESRAND, as well as for the additional 5 protocols considered in our study. Figure 5.5 supplies a high-resolution view of our results for the case where the number of agents is over 500, which appear in the marked rectangle in Figure 5.4. Note that a large number of stations (agents) represents high load and high interference since the portion of intersecting areas increases. Figure 5.6 shows the average standard deviation of the various protocols.

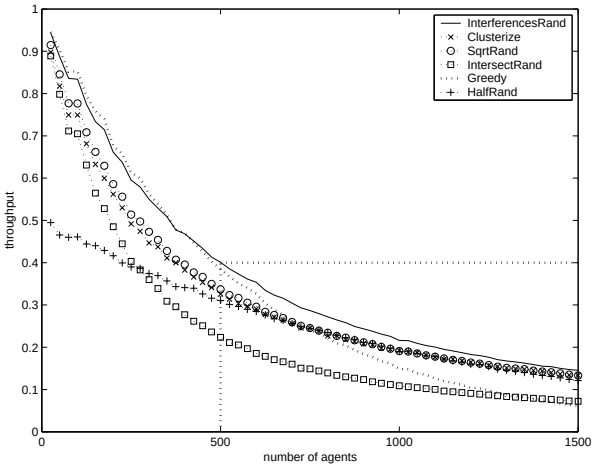


Figure 5.4: Comparison of the throughput of all 6 protocols.

As can be seen, our new protocol INTERFERENCESRAND, and the GREEDY protocol, demonstrate the best performance for moderate values of n. For higher loads, the performance of INTERFERENCESRAND is considerably better than the performance of all other protocols. This transition has been noted in several other smaller scale simulations we have conducted. It appears that the transition point is roughly the number of agents for which the overall area of the discs (n times π) equals the dimension of the domain. In other words, as the sum of transmission areas increases beyond the area of the domain, the policy implemented by our new protocol, INTERFERENCESRAND, of being aggressive in proportion with the actual amount of interference one suffers from one's neighbors, is the best strategy. On the other hand, for a relatively small number of agents, it appears that being greedy and constantly transmitting is the best strategy.

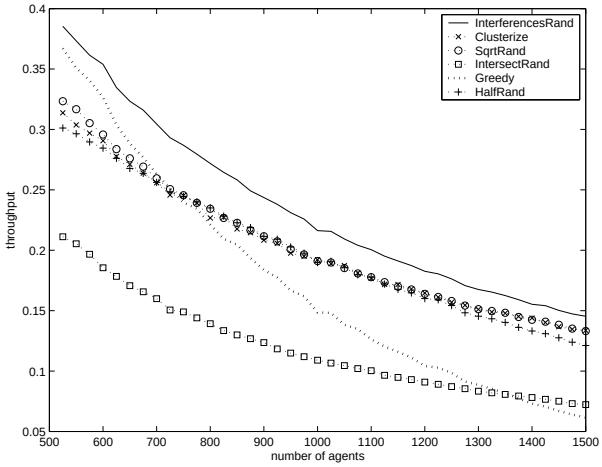


Figure 5.5: High resolution view of results for high-loads.

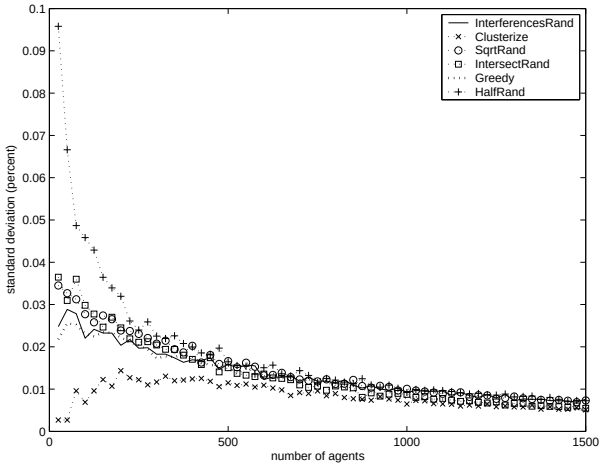


Figure 5.6: Average standard deviation percentage of the protocols for different number of agents.

This is probably due to the relatively small average value of $\alpha_{i,j}$, when the domain area is much larger than the number of agents (which is proportional to the area covered by them). Unlike the other protocols, which either exhibit good performance for a small number of agents, or for a large number of agents, the INTERFERENCESRAND protocol exhibited the best performance independent of the number of agents in the system. In this sense the protocol is more robust, and is clearly the best choice in cases where the system load varies. As can be seen in Figure 5.6, all protocols (except HALFRAND for small values of n) have very small standard deviation, which provides some confidence as to their performance for varying number of agents.

5.5 Discussion

We present a generalization of the classic multiple access model, by introducing the notion of soft collisions, which enable different transmissions to succeed simultaneously. This new

model captures the fact that collisions are a phenomenon experienced by the receiving end of the transmissions, and it depends on the amount of interferences sensed by this receiver from the various simultaneous transmissions.

We show that for homogeneous interferences, if agents are selfish, then the system's performance at equilibrium can be up to an exponential factor far away from the optimal performance. We further introduce a penalty function to be cast on the agents, that induces a much better performance in an equilibrium, which is at most a factor of e from the optimal performance. We then devise a protocol for non-homogeneous interferences and show using extensive simulation that it outperforms all other protocols for the problem.

Several interesting questions remain open. First, it would be interesting to see if penalizing an agent solely depending upon his chosen strategy, results in a sub-exponential price of anarchy, for the case of homogeneous interferences in the case where $\alpha \geq 2/e$. Another major goal is, of course, obtaining analytic guarantees as to the price of anarchy and the price of stability for non-homogeneous interferences. Another interesting question is to examine to role of receiver-feedback, which might enable the selfish user to better choose its strategy, resulting in a possible improvement of both the agent's own utility, as well as the overall network performance. Finally, our new model might also be applicable to other, non-wireless environments, where the phenomena of collisions is dominated by inter-agents interferences.

Bibliography

[1] Micah Adler, Sanjeev Khanna, Rajmohan Rajaraman, and Adi Rosén. Time-Constrained Scheduling of Weighted Packets on Trees and Meshes. *Algorithmica*, 36(2):123–152, 2003.

[2] Micah Adler, Arnold L. Rosenberg, Ramesh K. Sitaraman, and Walter Unger. Scheduling Time-Constrained Communication in Linear Networks. *Theoretical Computer Science*, 35(6):599–623, 2002.

[3] Ran Adler and Yossi Azar. Beating the Logarithmic Lower Bound: Randomized Preemptive Disjoint Paths and Call Control Algorithms. *Journal of Scheduling*, 6(2):113–129, 2003.

[4] William Aiello, Yishay Mansour, S. Rajagopolan, and Adi Rosén. Competitive Queue Policies for Differentiated Services. *Journal of Algorithms*, 55(2):113–141, 2005.

[5] William Aiello, Rafail Ostrovsky, Eyal Kushilevitz, and Adi Rosén. Dynamic Routing on Networks with Fixed-Sized Buffers. In *Proceedings of the 14th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 771–780, 2003.

[6] Susanne Albers and Markus Schmidt. On the Performance of Greedy Algorithms in Packet Buffering. *SIAM Journal on Computing*, 35(2):278–304, 2005.

[7] Eitan Altman, Dhiman Barman, Rachid El-Azouzi, and Tania Jiménez. A Game Theoretic Approach for Delay Minimization in Slotted Aloha. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3999–4003, 2004.

[8] Eitan Altman, Nicolas Bonneau, and Mérouane Debbah. Correlated Equilibrium in Access Control for Wireless Communications. In *Proceedings of the 5th International IFIP-TC6 Networking Conference (NETWORKING)*, pages 173–183, 2006.

[9] Eitan Altman, Rachid El-Azouzi, and Tania Jiménez. Slotted Aloha as a Game with Partial Information. *Computer Networks*, 45:701–713, 2004.

[10] Nir Andelman and Yishay Mansour. Competitive Management of Non-preemptive Queues with Multiple Values. In *Proceedings of the 17th International Symposium on Distributed Computing (DISC)*, pages 166–180, 2003.

[11] Stanislav Angelov, Sanjeev Khanna, and Keshav Kunal. The Network as a Storage Device: Dynamic Routing with Bounded Buffers. In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 1–13, 2005.

[12] Elliot Anshelevich, Anirban Dasgupta, Éva Tardos, and Tom Wexler. Near-optimal Network Design with Selfish Agents. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 511–520, 2003.

[13] The ATM Forum. *Traffic Management Specification*, March 1999. Version 4.1, AF-TM-0121.000.

[14] Baruch Awerbuch, Yossi Azar, Amos Fiat, Stefano Leonardi, and Adi Rosén. On-Line Competitive Algorithms for Call Admission in Optical Networks. *Algorithmica*, 31(1):29–43, 2001.

[15] Yossi Azar and Arik Litichevskey. Maximizing Throughput in Multi-Queue Switches. *Algorithmica*, 45(1):69–90, 2006.

[16] Yossi Azar and Yossi Richter. An Improved Algorithm for CIOQ Switches. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA)*, pages 65–76, 2004.

[17] Yossi Azar and Yossi Richter. Management of Multi-Queue Switches in QoS Networks. *Algorithmica*, 43(1-2):81–96, 2005.

[18] Yossi Azar and Rafi Zachut. Packet Routing and Information Gathering in Lines, Rings and Trees. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, pages 484–495, 2005.

[19] Sanjoy K. Baruah, Gilad Koren, D. Mao, B. Mishra, Arvind Raghunathan, Louis E. Rosier, Dennis Shasha, and Fuxing Wang. On the competitiveness of on-line real-time task scheduling. *Real Time Systems*, 4(2):125–144, 1992.

[20] Martin Beckmann, C. B. McGuire, and Christopher B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[21] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[22] Andrew Brzezinski, Gil Zussman, and Eytan Modiano. Enabling Distributed Throughput Maximization in Wireless Mesh Networks: A Partitioning Approach. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 26–37, 2006.

[23] George Christodoulou, Elias Koutsoupias, and Akash Nanavati. Coordination Mechanisms. In *Proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 345–357, 2004.

[24] Richard Cole, Yevgeniy Dodis, and Tim Roughgarden. Pricing Network Edges for Heterogeneous Selfish Users. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 521–530, 2003.

[25] comScore Networks. comScore releases August U.S. Video Metrix rankings, October 2006. Available online at: `http://www.comscore.com/press/release.asp?press=1035`.

[26] José R. Correa, Andreas S. Schulz, and Nicolás E. Stier. Selfish Routing in Capacitated Networks. *Mathematics of Operations Research*, 29(4):961–976, 2004.

[27] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). Internet RFC 3246, March 2002.

[28] Leah Epstein and Rob Van Stee. Buffer Management Problems. *ACM SIGACT News*, 35(3):58–66, September 2004.

[29] Amos Fiat, Yishay Mansour, and Uri Nadav. Efficient Contention Resolution for Selfish Agents. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[30] Lisa Fleischer, Kamal Jain, and Mohammad Mahdian. Tolls for heterogeneous selfish users in multicommodity networks and generalized congestion games. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 277–285, 2004.

[31] Cédric Florens, Massimo Franceschetti, and Robert J. McEliece. Lower Bounds on Data Collection Time in Sensory Networks. *IEEE Journal on Selected Areas in Communications*, 22(6):1110–1120, 2004.

[32] Juan A. Garay, Inder S. Gopal, Shay Kutten, Yishay Mansour, and Moti Yung. Efficient On-Line Call Control Algorithms. *Journal of Algorithms*, 23(1):180–194, 1997.

[33] Juan A. Garay, Joseph Naor, Bülent Yener, and Peng Zhao. On-line Admission Control and Packet Scheduling with Interleaving. In *Proceedings of the IEEE INFOCOM'02*, pages 94–103, New York, NY, June 2002.

[34] S.J. Golestani. A Stop-and-Go Queueing Framework for Congestion Management. In *ACM SIGCOMM*, pages 8–18, 1990.

[35] Eyal Gordon and Adi Rosén. Competitive Weighted Throughput Analysis of Greedy Protocols on DAGs. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 227–236, 2005.

[36] David Hay and Gabriel Scalosub. Jitter Regulation for Multiple Streams. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, pages 496–507, 2005.

[37] Youngmi Jin and George Kesidis. Equilibria of a Noncooperative Game for Heterogeneous Users of an ALOHA Network. *IEEE Communications Letters*, 6(7):282–284, 2002.

[38] C.R. Kalmanek, H. Kanakia, and S. Keshav. Rate Controlled Servers for Very High-Speed Networks. In *Proceedings of the Conference on Global Communications (GLOBECOM)*, pages 12–20, 1990.

[39] S. Keshav. *An Engineering Approach to Computer Networking*. Addison-Wesley Publishing Co., 1997.

[40] I. Keslassy, M. Kodialam, T.V. Lakshman, and D. Stiliadis. On guaranteed smooth scheduling for input-queued switches. *IEEE/ACM Transactions on Networking*, 13(6):1364–1375, December 2005.

[41] Alexander Kesselman, Zvi Lotker, Yishay Mansour, and Boaz Patt-Shamir. Buffer Overflows of Merging Streams. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA)*, pages 349–360, 2003.

[42] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer Overflow Management in QoS Switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[43] Alexander Kesselman, Yishay Mansour, and Rob van Stee. Improved Competitive Guarantees for QoS Buffering. *Algorithmica*, 43(1-2):63–80, 2005.

[44] Alexander Kesselman and Adi Rosén. Scheduling policies for CIOQ switches. *Journal of Algorithms*, 60(1):60–83, 2006.

[45] H. Koga. Jitter Regulation in an Internet Router with Delay Constraint. *Journal of Scheduling*, 4(6):355–377, 2001.

[46] Gilad Koren and Dennis Shasha. D^{over}: An Optimal On-Line Scheduling Algorithm for Overloaded Uniprocessor Real-Time Systems. *SIAM Journal on Computing*, 24(2):318–339, 1995.

[47] Kishore Kothapalli, Melih Onus, Andréa Richa, and Christian Scheideler. Efficient Broadcasting and Gathering in Wireless Ad-Hoc Networks. In *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 346–351, 2005.

[48] Kishore Kothapalli and Christian Scheideler. Information Gathering in Adversarial Systems: Lines and Cycles. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 333–342, 2003.

[49] Anis Koubâa, Màrio Alves, Melek Attia, and Anneleen Van Nieuwenhuyse. Collision-Free Beacon Scheduling Mechanisms for IEEE 802.15.4/Zigbee Cluster-Tree Wireless Sensor Networks. In *Proceedings of the 7th International Workshop on Applications and Services in Wireless Networks (ASWN)*, 2007.

[50] Elias Koutsoupias and Christos Papadimitriou. Worst-Case Equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 404–413, 1999.

[51] Jörg Liebeherr. Multimedia Networks: Issues and Challanges. *IEEE Computer*, 28(4):68–69, 1995.

[52] Richard J. Lipton and Andrew Tomkins. Online Interval Scheduling. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 302–311, 1994.

[53] Richard T. B. Ma, Vishal Misra, and Dan Rubenstein. Modeling and Analysis of Generalized Slotted-Aloha MAC Protocols in Cooperative, Competitive and Adversarial Environments. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, page 62, 2006.

[54] Allen B. MacKenzie and Stephen B. Wicker. Selfish Users in Aloha: a Game-Theoretic Approach. In *Proceedings of the IEEE VTS 53rd Vehicular Technology Conference (VTC)*, pages 1354–1357, Fall 2001.

[55] Allen B. MacKenzie and Stephen B. Wicker. Stability of Multipacket Slotted Aloha with Selfish Users and Perfect Information. In *Proceedings of IEEE INFOCOM 2003, The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1583–1590, 2003.

[56] Y. Mansour and B. Patt-Shamir. Jitter Control in QoS Networks. *IEEE/ACM Transactions on Networking*, 9(4):492–502, August 2001.

[57] Peter Marbach and Ran Pang. Transmission Costs, Selfish Nodes, and Protocol Design. In *Proceedings of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt)*, pages 31–40, 2005.

[58] Ishai Menache and Nahum Shimkin. Fixed-Rate Equilibrium in Wireless Collision Channels. In *Proceedings of the 1st International Conference on Network Control and Optimization (Net-Coop)*, 2007. To appear.

[59] Eytan Modiano, Devavrat Shah, and Gil Zussman. Maximizing throughput in wireless networks via gossiping. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance)*, pages 27–38, 2006.

[60] Joseph (Seffi) Naor, Danny Raz, and Gabriel Scalosub. Soft Collisions in Wireless Networks. Submitted.

[61] Joseph (Seffi) Naor, Adi Rosén, and Gabriel Scalosub. Online Time-Constrained Scheduling in Linear Networks. In *Proceedings of IEEE INFOCOM 2005, The 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 855–865, 2005.

[62] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[63] C. Partridge. Isochronous Applications Do Not Require Jitter-Controlled Networks. Internet RFC 1257, September 1991.

[64] Jennifer Rexford, John Hall, and Kang G. Shin. A Router Architecture for Real-Time Communication in Multicomputer Networks. *IEEE Transactions on Computers*, 47(10):1088–1101, 1998.

[65] Adi Rosén and Gabriel Scalosub. Rate vs. Buffer Size: Greedy Information Gathering on the Line. In *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 305–314, 2007.

[66] Tim Roughgarden and Éva Tardos. How Bad is Selfish Routing? *Journal of the ACM*, 49(2):236–259, 2002.

[67] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28(2):202–208, 1985.

[68] A.S. Tanenbaum. *Computer Networks*. Prentice Hall, fourth edition, 2003.

[69] Mario Čagalj, Saurabh Ganeriwal, Imad Aad, and Jean-Pierre Hubaux. On Selfish Behavior in CSMA/CA Networks. In *Proceedings of IEEE INFOCOM 2005, The 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2513–2524, 2005.

[70] D.C. Verma, H. Zhang, and D. Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proceedings of TriComm*, pages 35–46, 1991.

[71] Dandan Wang, Cristina Comaniciu, and Uf Tureli. A Fair and Efficient Pricing Strategy for Slotted Aloha in MPR Models. In *Proceedings of the IEEE VTS 64th Vehicular Technology Conference (VTC)*, pages 1–5, Fall 2006.

[72] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet Switching Networks. *Proceedings of the IEEE*, 83(10):1374–1396, October 1995.

[73] H. Zhang and D. Ferrari. Rate-Controlled Service Disciplines. *Journal of High-Speed Networks*, 3(4):389–412, 1994.

המערכת המושג בזכות מנגנון הטלת הקנסות. תוצאה זו מראה כי מחיר היציבות של המערכת עם קנסות הוא קבוע. יתר על כן, אנו מראים שבמקרה שבו ההפרעות ההדדיות בין השחקנים השונים אינן גבוהות מדי, אזי זהו שיווי המשקל היחיד, כלומר גם מחיר האנרכיה במקרה זה הוא קבוע. עבור הפרעות שאינן הומוגניות, אנו מפתחים פרוטוקול חדש לבעיה ומראים באמצעות סימולציה נרחבת כי ביצועיו עולים על ביצועי פרוטוקולים אחרים מקובלים לבעיה. יתר על כן, הפרוטוקול שלנו איננו רגיש לעומס על המערכת וביצועיו אינם תלויים במספר התחנות המשדרות.

מאוחר יותר בצורה מחזורית ככל שניתן. בעיה זו היא מן הבעיות הבסיסיות ביותר ברשתות בהן יש דרישה להבטחת איכות שירות והיא מהותית בעיקר ברשתות התומכות בשידורי מולטימדיה ושידורי קול ווידאו מקוונים. אנו מציגים אלגוריתם פולינומי אשר פותר את הגירסה הלא-מקוונת של הבעיה באופן אופטימלי. לגבי הגרסא המקוונת של הבעיה אנו מראים כי ניתן להשתמש באלגוריתם קיים עבור שידור יחיד, העושה שימוש בחוצץ בגודל $2B$ ולבצע הסדרת שידור של M שידורים במקביל, תוך שימוש בחוצץ בגודל כולל של $2MB$. בצורה זו ניתן להבטיח כי הרעד המקסימלי בכל אחד מן השידורים לא יהיה גבוה מן הרעד האופטימלי שניתן להשיג עם חוצץ בגודל כולל B. אנו מראים כי הגדלת משאבים כזו היא הכרחית על ידי הצגת חסם תחתון לפיו ניתן לגרום לכל אלגוריתם העושה שימוש בחוצץ קטן מ-$\Omega(MB)$ להשיג רעד גדול כרצוננו, אפילו כאשר הרעד האופטימלי האפשרי הוא אפס. תוצאה זו מראה כי גם כאשר המטרה היא למזער את הרעד הממוצע יש הכרח בהגדלת משאבים פרופורציונית למספר השידורים.

בעיה נוספת המוצגת בעבודה זו היא בעית איסוף מידע על הישר המכוון. בבעיה זו ישנו יריב המזריק חבילות לצמתים שונים בישר, כאשר בכל צומת יש חוצץ בעל גודל מוגבל. בכל יחידת זמן כל צומת יכול לבחור אם להעביר חבילה המצויה בחוצץ שלו לצומת הבא על הישר, או להמנע מכך. המטרה היא להעביר כמה שיותר חבילות לתחנת הקצה, כאשר במקרה שהיריב מזריק חבילות לצומת שבו החוצץ מלא, חבילות אובדות. אנו מרחיבים מודל קודם שטיפל בבעיה ומגדירים משפחות של יריבים הנבדלות זו מזו בקצב ההזרקה המקסימלי של היריב. במסגרת זו אנו מראים כי הפרוטוקול החמדן, אשר מקבל חבילות חדשות כאשר יש לו מקום בחוצץ ושולח חבילה אם החוצץ איננו ריק, משיג יחס תחרותיות שונה בהתאם לפרמטרים השונים של המערכת, כגון גודל הרשת, קצב ההזרקה של היריב וגודל החוצץ בכל צומת. להבדיל מתוצאות קודמות אשר הבטיחו יחס תחרותיות התלוי בגודל הרשת בלבד, התוצאות שלנו מדגישות את החשיבות שיש לגודל החוצץ בביצועי המערכת וביחס התחרותיות שהפרוטוקול יכול להבטיח בתנאים השונים. יתר על כן, עבור משפחות מסויימות של יריבים, הקצאת משאבים מספיקה מבחינת גודל החוצצים בצמתי הרשת מבטיחה ביצועים אופטימליים לפרוטוקול, בניגוד למצב בו הקצאת המשאבים איננה מספקת והביצועים עלולים להיות רחוקים מאד מן האופטימום.

הבעיה האחרונה בה אנו דנים בעבודה זו היא בעית הגישה המשותפת ברשתות אלחוטיות עם התנגשויות מוגבלות הנובעות מקרבה של התחנות המשדרות ברשת. אנו ממדלים את הבעיה כמשחק לא שיתופי, שבו כל תחנה היא שחקן המעוניין למקסם את התועלת שלו. הרווחה החברתית נמדדת על ידי מספר השידורים המוצלחים בו זמנית של כל השחקנים, כאשר לשחקנים המשדרים סימולטנית יש השפעה הרסנית אחד על השני, הנמדדת על ידי פרמטר הפרעה הדדי. השפעה זו מקטינה את סיכויי הצלחת השידור של שחקן אם יש שחקנים אחרים המשדרים עמו בו זמנית. למקרה שבו כל ההפרעות הן הומוגניות, כלומר מקרה שבו כל שני שחקנים מפריעים זה לזה באותה מידה, אנו מנתחים את שיווי משקל נאש של המשחק ומראים שכאשר כל שחקן מעוניין למקסם את הסתברות ההצלחה של שידורו ישנו שיווי משקל יחיד. היחס בין הביצועים האופטימליים של המערכת לבין ביצועיה בשיווי משקל זה יכול להיות אקספוננציאלי במספר התחנות המשדרות. תוצאה זו מראה כי מחיר האנרכיה, כמו גם מחיר היציבות של המערכת יכול להיות אקספוננציאלי. אנו מציגים מנגנון של הטלת קנסות על התחנות המשדרות בו גובה הקנס הוא ביחס ישר להסתברות השידור של התחנה. עבור מנגנון זה אנו מראים כי ישנו שיווי משקל נאש שבו היחס בין הביצועיים האופטימליים לבין הביצועים בשיווי המשקל חסומים על ידי קבוע, כלומר שיפור אקספוננציאלי בביצועי

תקציר

רשתות תקשורת נתונים מהוות מזה עשורים רבים מרכיב חשוב בחיים המודרנים. המורכבות ההולכת וגדלה של רשתות אלו, הן בהבטי ארכיטקטורה והן מבחינת השירותים אשר רשתות אלו אמורות לספק, מעלה צורך רב בפיתוח אלגוריתמים ופרוטוקולים חדשים אשר יכולים להבטיח רמת ביצועים גבוהה. פרוטוקולים אלו צריכים להתמודד עם קצבי תעבורה גבוהים ולהבטיח רמת שירות נאותה לסוגי התעבורה השונים. בעבודה זו אנו עוסקים במספר בעיות בסיסיות ברשתות תקשורת נתונים מתקדמות, תוך התמקדות במצבים בהם יש למערכת מגבלות שונות. סוג אחד של מגבלות עמן אנו מתמודדים נובע ממצבים בהם ישנו העדר מידע בדבר ארועים עתידיים. סביבה כזו מכונה *סביבה מקוונת*. סוג שני של מגבלות עמן אנו מתמודדים נובע ממצבים בהם ישנו העדר תאום ושיתוף פעולה בין המרכיבים השונים של המערכת. בסביבה זו אנו מנתחים את ביצועי המערכת תוך שימוש בכלים מתורת המשחקים.

אנו מטפלים בבעיות השונות מנקודת המבט של *אנליזה תחרותית* ומפתחים אלגוריתמים ופרוטוקולים לבעיות השונות מתוך הסתכלות זו. השימוש באנליזה תחרותית מאפשרת לאלגוריתמים ולפרוטוקולים השונים המוצעים כאן להיות שימושיים במגוון רחב של מערכות, ללא כל תלות בהנחות מוקדמות על סוגי התעבורה או בהתפלגויות שונות על פיה מיוצרת התעבורה. אלגוריתם מקוון לבעית מקסימיזציה הוא c תחרותי אם לכל קלט אפשרי, היחס בין ביצועיו לבין הביצועים האופטימליים האפשריים הוא לפחות c.

הבעיה הראשונה עמה אנו מתמודדים היא ניתוב ותזמון ברשתות אופטיות, כאשר לכל חבילה יש מועד תפוגה אשר עד אליו היא צריכה להגיע ליעדה. בכדי להמנע מהמרות מיותרות מן התווך האופטי לתווך האלקטרוני ובשל הקושי במימוש חוצצים בתווך האופטי, חבילות המתחילות לנוע ברשת חייבות להמשיך בנתיבן ללא עיכובים. בעבודה זו אנו מציגים לראשונה אלגוריתמים מקוונים לבעיה, ומתמקדים ברשתות שבהן הטופולוגיה היא ישר או טבעת. למקרה שבו לכל החבילות אותו ערך והמטרה היא להביא למקסימום את מספר החבילות המגיעות ליעדן לפני מועד תפוגתן, אנו מציגים אלגוריתם המבטיח יחס תחרותיות לוגריתמי בגודל הרשת. כאשר הרווח מכל חבילה פרופורציוני לאורך המסלול שאותה צריכה החבילה לעבור, אנו מציגים אלגוריתם בעל יחס תחרותיות $2\phi+1\sim4.23$, כאשר ϕ הוא יחס הזהב, וחסם תחתון של 4 על יחס התחרותיות של כל אלגוריתם דטרמיניסטי לבעיה. במקרה שבו אין כל מידע על הרווח מכל חבילה, אנו מציגים אלגוריתם בעל יחס תחרותיות של β, כאשר β הוא היחס בין הצפיפות המקסימלית של חבילה כלשהי במערכת לבין זו המינימלית. הצפיפות של חבילה מוגדרת להיות היחס בין הרווח המובטח מהבאתה ליעדה טרם זמן תפוגתה לבין אורך המסלול שאותה צריכה החבילה לעבור.

בעיה נוספת בה אנו מטפלים היא הצורך בהבטחת רעד (Jitter) נמוך עבור מספר רב של שידורים תוך שימוש בחוצץ משותף. בבעיה זו אנו אמורים להסדיר את התעבורה המתקבלת, שהיא שילוב של מספר שידורים, על ידי צבירת חבילות בחוצץ בעל גודל מוגבל ושליחתן

המחקר נעשה בהנחיית פרופ׳ ספי נאור, פרופ׳/ח דני רז ודר. עדי רוזן בפקולטה למדעי המחשב.

ברצוני להודות למנחי, אשר סייעו לי וכיוונו אותי לאורך הדרך הפתלתלה. באישיותכם וברוחכם כאנשי מחקר הייתם לי מקור השראה במהלך השנים, ואני מודה שהתאפשר לי לזכות לעבוד עמכם, ולשרות תחת כנפיכם והדרכתכם. ספי, עבור העידוד והסיוע התמידי, החיפוש אחר בעיות חדשות ורעיונות חדשים, ועבור העצה החמה והחזרתי לדרך הישר בשעות אי הוודאות. עדי, עבור הקדשת ימים ארוכים בבדיקת רעיונות ומציאת פתרונות, וגם סתם שיחות על החיים, ועל הקפדנות והדרישה להוציא מעצמי את המיטב, במדוייק, ובאלגנטיות. דני, על העונג שהיה לי לעבוד איתך, על תשומת הלב, הדאגה, והתמיכה, ועל העובדה שדלתך תמיד היתה פתוחה על מנת לאפשר לי לגלגל רעיונות, ולבדוק אפשרויות.

ברצוני להודות לדוד חי, דודו אמזלג וראובן בר-יהודה עמם התמזל מזלי לשתף פעולה בעבודה משותפת.

ברצוני להודות לחברי הרבים בטכניון, ומחוצה לו, אשר היוו מקור תמידי לתמיכה וסיוע.

ברצוני להודות לאמי ולאחי, אשר תמיד היו שם בשבילי, בכל עת מחפשים את הדרך לסייע ולתת לי את הזמן ואת האפשרות להתמקד בעבודתי. אהבתכם ותמיכתכם היו, ותמיד תהינה, שמורות עמי.

לבסוף, ברצוני להודות לבת זוגי, דינה, שתמכה בי לאורך תקופה ארוכה של עבודה אינטנסיבית, מתח וקשיים. לא הייתי יכול לעשות זאת בלעדי אהבתך והדחיפה המתמדת.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

בעיות ניתוב ותזמון ברשתות תקשורת

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

גבריאל סקלוסוב

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

חשוון התשס״ח חיפה אוקטובר 2007

בעיות ניתוב ותזמון ברשתות תקשורת

גבריאל סקלוסוב