Stability in Adversarial Queueing Theory

by

Panagiotis Tsaparas

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

A queueing network is a set of interconnected servers. Customers are injected continuously in the system, inducing some workload for each server. A fundamental issue that arises in this context is that of stability: will the total workload in the system remain bounded over time? In this thesis, we investigate the question of stability within the model defined in [BKR+96], where workload is induced by an adversary. Prior work in the adversarial model considers only packet routing networks, a special case of queueing networks, where all customers require unit service time. We extend this model to include more general queueing networks, where the customers may require different service times at different servers of the network. We show that the queueing policies proven to be universally stable for packet routing networks, are universally stable in the new adversarial model as well. Unlike the packet routing networks, we prove that in the new model, the unidirectional ring is not universally stable. Furthermore, we define four new queueing policies for the new model, which depend upon the service times of the customers, and we examine their stability. Finally, following the outline in [BKSW96], we provide a detailed proof that the Nearest-To-Go queueing policy can become unstable when workload is injected at arbitrarily small constant rate.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Problem

Queueing networks are important models of complex systems, such as packet routing networks, manufacturing systems, and time shared computer systems. In this work we analyze the behavior of queueing networks where customers are continuously injected into the system. Each customer induces some workload in the network. A crucial question arising in this context is that of **stability** — will the total workload in the system remain bounded as the system runs for an arbitrarily long period of time? The answer to this question typically depends on the **rate** at which workload is injected into the system and the **queueing policy** that is used at the nodes of the network.

A queueing network is a set of interconnected **servers**. We assume that each server is associated with a single queue of infinite size, and each queue is serviced by a single server. Customers arrive at the servers of the network. When the service of some customer at some server is completed the customer moves to another server or it exits the system. If more than one customer wishes to be serviced at some server S, the queueing policy chooses one of the customers to be serviced next; the remainder of these customers wait in the queue of server S. A queueing policy is **work-conserving** if a server is never idle whenever there is at least one customer in the queue of the server. A queueing policy is called **non-preemptive** if the service of some customer at some server is never interrupted until it is completed.

In general, a queueing network is characterized by the following parameters.

- The process that injects the customers in the network.
- The routing of the customers in the network.
- The service times of the customers at the servers.
- The queueing policy.

It is obvious that if the rate at which workload arrives at some server is greater than the rate at which the workload is serviced at that server, then the system becomes unstable. A natural condition for stability, referred to as the **rate condition**, is that the rate at which workload is induced at any server is strictly less than the service rate of the server. Intuitively, it seems that the rate condition should be sufficient for stability of any queueing network. This is true for many systems [Kle75, Kel79], but it is not true in all cases [LK91, RS92, DW96, Bra94a, Bra94b, Sei94]. The question of necessary and sufficient conditions for stability is one of the most interesting questions in the area of queueing networks.

Queueing networks are the object of study of queueing theory. Typical assumptions in queueing theory are that the customers are injected according to a time invariant distribution (often a Poisson process), and that the service time of a customer, or customer type at some server is a random variable following a time invariant distribution (often an exponential distribution). In this thesis we work within a model proposed by Borodin et al. [BKR+96], in which probabilistic assumptions are replaced by worst-case inputs. The underlying goal is to determine whether it is feasible to prove stability results even when customers are injected by an **adversary**, rather than a randomized process; the adversary determines the route of the customers in the system and the service time they receive at each server. For systems that are not stable, we seek to establish instability within the weakest possible adversarial model.

1.2 Adversarial Queueing Theory

Adversarial Queueing Theory was introduced by Borodin et al. [BKR+96], for the study of packet routing networks. A packet routing network is a special case of a queueing network where packets (i.e. customers) move on the edges (i.e. servers) of a directed graph. All packets take unit time to cross any edge in the network. Any packet routing network

can be easily transformed into a queueing network, simply by transforming edges into servers. The resulting queueing network is stable, if and only if the packet routing network is stable. However the converse does not hold; there exist queueing networks that can not be transformed into packet routing networks.

In the adversarial model the packets are injected by an **adversary**, rather than some independent randomized process. The model considers time evolution in the system as a game between an adversary and a queueing policy. Time proceeds in discrete steps. In each time step, the adversary injects a set of packets at some nodes; for each packet it specifies a path of edges that it must traverse, after which the packet is absorbed. A queueing policy is said to be stable against an adversary if the total number of packets in the system is bounded over time.

A crucial parameter of the adversary is the rate. In [BKR+96] a request by the adversary is defined to be a set of packets requesting edge-disjoint paths; in their terminology an adversary injects at rate r, if for all t, no more than $\lceil rt \rceil$ requests are made in any interval of t steps. We call this model the **adversarial path-packing model**. A more general model was suggested in [BKR+96], and was fully developed in [AAF+96], in which the rate of the adversary is specified by a pair (w, r), where w is a natural number, and $0 < r < 1$. The requirement of the adversary is the following: of the packets that the adversary injects in any interval of w steps at most $\lfloor rw \rfloor$ can have paths that contain any one edge. We will refer to this model as the **adversarial packet routing model**.

In [AAF+96] a **universally stable queueing policy** is defined as a queueing policy that is stable against any adversary of rate less than 1, on every network. A **universally stable graph** is defined as a graph G, such that every queueing policy is stable against every adversary of rate less than 1 on G.

Borodin et al. [BKR+96] showed that directed acyclic graphs (DAGs) are universally stable in the adversarial packet routing model. In [AAF+96] the unidirectional ring is shown to be universally stable in the same model. Furthermore, Andrews et al., prove that in this model the queueing policies **Furthest-To-Go (FTG)**, **Longest-In-System (LIS)** and **Shortest-In-System (SIS)** are universally stable. The SIS queueing policy gives precedence to the customer most recently injected in the network; the LIS queueing policy gives precedence to the customer injected the earliest in the network; the FTG queueing policy gives precedence to the customer whose distance to its destination is maximal. However, two

of these queueing policies (FTG and SIS) can require queues of exponential size in the size of the network, while for LIS the best known upper bound on the queue size is also exponential. Andrews et al., show that there is a distributed randomized protocol with a polynomial bound on the queue size, with high probability. On the negative side, they present instability examples for the queueing policies **First-In-First-Out (FIFO)**, **Last-In-First-Out (LIFO)** and **Nearest-To-Go (NTG)**. The NTG queueing policy gives precedence to the packet whose distance to its destination is minimal. They prove that these protocols can be unstable on commonly used networks, including arrays and hypercubes.

1.3 Our results

The model presented in [BKR+96, AAF+96] is directed to the study of packet routing networks, where customers require unit service times. In this thesis we define a broader class of adversaries that models general queueing networks. This model was first suggested in [BKR+96]. In the new model the adversary injects customers that may require different service times at each server along their path. The rate of an adversary in our work will be specified by a pair (w, μ), where w is any number and $0 < \mu < 1$. The requirement on the adversary is the following: the total service requirement that the adversary induces at any server, in any interval of w units of time is at most μw. We will refer to this new model as the **adversarial queueing model**.

The general adversarial model allows for some new queueing policies that depend upon the service requirements of the customers. We consider the following queueing policies:

- **Most-Time-To-Go (MTTG) :** The MTTG queueing policy gives precedence to the customer whose remaining service time in the system is maximal.
- **Least-Time-To-Go (LTTG):** The LTTG queueing policy gives precedence to the customer whose remaining service time in the system is minimal.
- **Most-Service-Demand (MSD):** The MSD queueing policy gives precedence to the customer that requires the maximum service time at a given server among those queued for that server.
- **Least-Service-Demand (LSD):** The LSD queueing policy gives precedence to the customer that requires the minimum service time at a given server among those that

are queued at that server.

We will prove that the queueing policies FTG, LIS and SIS remain universally stable in the adversarial queueing model. Furthermore, the queueing policy MTTG is universally stable against a restricted class of adversaries, where the service requirements of any customer injected by the adversary, at any server, are bounded from below. If we allow the adversary to inject customers that require arbitrarily small service times at some servers, then we will show that MTTG can be made unstable. The new model, and in particular the stability results, can be motivated by packet routing networks, where packets can have different sizes, and therefore take different times to cross an edge, or the edges themselves can transmit at different rates.

In the adversarial queueing model the unidirectional ring is not universally stable. We will show that there are simple queueing policies, such as NTG, MSD, LTTG and FIFO, for which there is an adversary of rate less than 1, that causes them to become unstable on the ring. This implies that any non-acyclic network, with a big enough cycle can be made unstable.

So far our focus has been on adversaries that inject customers at some rate less than 1. It is interesting to study the behavior of queueing policies against adversaries with arbitrarily small injection rate. A recent result of Borodin, Kleinberg, Sudan and Williamson [BKSW96] indicates that NTG can be unstable against adversaries of arbitrary small positive rates. We present a detailed proof of the instability of NTG on a packet routing network, when packets are injected at constant rate. The adversary determines only the initial configuration of the system. Depending on the size of the network the injection rate can become arbitrarily small. The instability example answers negatively an open question raised in [AAF+96]: is there some rate for which any queuing policy is universally stable?

The rest of the thesis is organized as follows. In Chapter 2 we present a short review of some relevant results in queueing theory. In Chapter 3 we introduce the new adversarial model, and present proofs of the stability and instability results. In Chapter 4 we present the proof of instability of NTG against an weaker "adversary" of arbitrarily small rate. The thesis concludes in Chapter 5 with a summary of the results and a discussion of some open problems.

Chapter 2

Previous Work

2.1 Classical Queueing Theory

Queueing theory is a well developed subject of fundamental importance to computer science, providing the mathematical foundation and the main analytical tools for performance evaluation of computer systems. In this section we give a short review of some basic results and sketch some of the most recent advances in the area of queueing theory. A more detailed survey can be found in [Ana96].

We begin by outlining some of the basic definitions and techniques (see, for example, the texts of Kelly [Kel79] and Kleinrock [Kle75]). Typically, in queueing theory a system is viewed as a Markov process in which the "state" of the system changes whenever there is an injection or a service completion. Here the state of the network is a vector describing the state of each server; the state of a server describes the customers currently waiting for service as well as the customer(s) being serviced and the remaining time of such service. We say that the queueing network is stable if this Markov process has a stationary distribution and call this the equilibrium state.

Many different models have been proposed for the study of queueing networks. Depending on the kind of customers that are injected in the system, we distinguish two classes of networks: **single-class** networks and **multiclass** networks.

2.1.1 Single-class networks

In single-class networks all customers that visit a particular server of the network are essentially indistinguishable. All customers receive the same service and follow the same routing scheme. Single-class networks can also be called **station-centered** networks, since the service times and the routing variables are associated with the servers, and they are handed out to customers as they are picked for service [BF94].

The model of single-class networks was first introduced by Jackson [Jac63]. A **Jackson network** consists of J interconnected servers. The outside world is conventionally indexed as server 0. The arrivals of customers from the external world to a server are all generated by an independent Poisson process. An arriving customer is routed to server j with probability p_{0j}, with $\sum_{j=1}^{J} p_{0j} = 1$, where it is queued in FIFO order. The service time required by a customer at server j is exponentially distributed with mean μ_j. Service times are independent and identically distributed and independent of other parameters. When the service of a customer at server j is completed, the customer moves to sever k with probability p_{jk}, or it leaves the system with probability p_{j0}, where $\sum_{k=0}^{J} p_{jk} = 1$. Consider now the n-th customer at server j. The server to which this customer is routed next is a random variable $r_j(n)$. Thus, we can view routing at server j as a random process $r_j = \{r_j(n),\ n \geq 1\}$ indexed by the customer number. The process r_j is independent and identically distributed and independent of the arrival process and the service times. This routing mechanism is often called Bernoulli routing. It is also referred to as Markovian routing, since the next server that a customer moves to depends only on the server at which it is currently serviced.

If the rate condition is satisfied, Jackson networks have a **product form** solution: the equilibrium state of the network is the product of the equilibrium states of the individual servers. Product form solutions can be derived for such networks with queueing policies other than FIFO, as long as the queueing policy is work conserving.

Consider now a direct generalization of the Jackson network, where the assumptions of Poisson arrivals and exponentially distributed service times are replaced by general interarrival and service time distributions. These networks are classed as **generalized Jackson networks**. The problem of determining the stability of generalized Jackson networks turned out to be more difficult to resolve. Satisfactory solutions have been obtained only recently by Borovkov [Bor86], Foss [Fos89, Fos91], Meyn and Down [MD94], Chang et al. [CTK94],

Baccelli and Foss [BF94, BFM96] and Dai [Dai95] for various assumptions on the arrival process and the service times. In particular, systems with Bernoulli routing and independent identically distributed inter-arrival and service times, are shown to be stable for any work-conserving, non-preemptive queueing policy. Some additional constraints may be required for the inter-arrival times. Typically it is assumed that the inter-arrival times are bounded and spread out. Baccelli and Foss [BF94] prove that if the routing at every server is a random sequence that is stationary and ergodic [BF94], and the arrival process and the service times are also stationary and ergodic, then the system is stable under any work-conserving, non-preemptive queueing policy.

2.1.2 Multiclass networks

In multiclass networks, customers that arrive at some server belong to different classes. The routing and possibly the service time of the customers at a server vary depending on the class of the customer. This model allows for more general routing schemes. Multiclass networks can also be called **customer-centered** networks since the routing variables and the service times are associated with the individual customers [BF94].

The multiclass model was introduced independently by Kelly [Kel75] and Basket et al. [BCMP75]. We first present the model described in [Kel75]. A **Kelly network** consists of J servers and I different types of customers. Customers of type i are injected by a Poisson process of rate $v(i)$ and pass through the predefined sequence of servers

$$r(i,1), r(i,2), \ldots, r(i,N_i),$$

and then exit the network. The class of a customer is defined as the type of the customer and the stage along its route that this customer has reached. Each customer requires an amount of service time at each server that is exponentially distributed with unit mean. The service of customers at any server j is defined by a set of rules. These rules are general enough to model common queueing policies such as FIFO, LIFO and Processor Sharing (PS). However, they can not model queueing policies that distinguish between customers of different classes. Kelly proved that this class of networks is stable under the rate condition, and it has a product form solution.

Another class of stable networks described in [Kel75, Kel79] is the following. The cus-

tomers are still injected by a Poisson process. The service time required by a customer at any server follows a distribution, that may be other than exponential, and may depend upon the class of the customer. Under the rate condition, this class of networks is stable, under LIFO and PS queueing policies, and has a product form solution.

A related model was introduced independently by Basket et al [BCMP75]. There are J servers and I classes of customers. Customers are injected by a Poisson process with arrival rate dependent on the state of the system. An arriving customer enters at server i in class r with a fixed probability q_{ir}. A customer of class r that completes its service at server i will next require service at server j in class s with a certain probability $P_{i,r:j,s}$. Basket et al., proved that under the rate condition the following classes of networks are stable and have a product form solution.

- The queueing policy is FIFO and all customers have exponentially distributed service times at each server.

- The queueing policy is LIFO or PS. Each class of customers may have a distinct service time distribution. The service time distributions have a rational Laplace transformation.

BCMP and Kelly networks assume Poisson arrivals and impose restrictions on the service times and the queueing policy. The question of stability of multiclass networks for general queueing policies, under generalized distributional assumptions on the arrivals and the service times has only been considered recently. Lu and Kumar [LK91] consider a special class of networks called "re-entrant lines" [Kum93]. In this model all customers are of the same type; they enter the system at the same server and follow the same route. The distinctive feature is that customers may revisit the same server more than once with different service requirements at each visit. The arrival process follows the **leaky-bucket** model introduced by Cruz [Cru91a, Cru91b]. The arrivals of customers are deterministic: during any interval of time $[s, t]$ the number of customers that arrive in the system is

$$\lambda(t - s) + \gamma,$$

where λ is the arrival rate, and γ is a constant that allows for some burstiness in the arrivals. Lu and Kumar prove that the queueing policies First-Buffer-First-Serve (FBFS)

and Last-Buffer-First-Serve (LBFS) are stable. FBFS gives priority to the customers that are in the earliest stage of their route, while LBFS gives priority to the customers that are in the latest stage of their route. These queueing policies are equivalent to the FTG and NTG queueing policies we defined for general networks.

Lu and Kumar also consider a generalization of the re-entrant lines, where there are more than one type of customer. They prove that in this setting some variations of FBFS and LBFS are stable.

Rybko and Stolyar [RS92] consider a two server network with two types of customers moving in opposite directions. They prove that the system with Poisson arrivals and arbitrary service times is stable under FIFO queueing policy. Their analysis, using fluid models, inspired the work of Dai [Dai95], where fluid models are used to prove the stability of more general networks. In [Dai95] inter-arrival and service times are independent and identically distributed. The inter-arrival times are assumed to be unbounded and spread out. The fluid model of a queueing network is a solution to a set of equations that model the network dynamics. Dai [Dai95] proves that a queueing discipline is stable if the corresponding fluid model is stable. It remains a major open question if the converse also holds. Using fluid models Dai and Weiss [DW96] prove that FBFS and LBFS are stable for any re-entrant line. Furthermore, the unidirectional ring of servers is stable under any work-conserving queueing policy.

Recent results by Bramson [Bra96] use the fluid model defined by Dai to prove the stability of multiclass networks under general distributions on the inter-arrival and service times. Bramson considers networks where the service time of a customer at a server depends on the server and not on the class of the customer that is being serviced. These networks are termed in [Bra96] as **generalized Kelly networks**. He proves that under the assumptions of the fluid model for inter-arrival and service times the system is stable under FIFO and PS queueing policies.

2.1.3 Instabilities in multiclass networks

For a very long time it seemed plausible that the rate condition could be sufficient for stability of general networks. This belief was recently shattered by a series of instability examples. We now briefly sketch the genesis of these examples.

Consider a two server network, with servers S_1 and S_2. There are two buffers at each

server; B_1 and B_4 in S_1, and B_2 and B_3 in S_2. Let m_1, m_2, m_3, and m_4 be the service times required by any customer at each buffer. We assume that every server works at rate 1. There is a single type of customers in the system that follows the sequence B_1, B_2, B_3, B_4. Customers arrive at constant rate; one customer arrives at each time step. Kumar and Seidman [KS90] consider the special case of a manufacturing system. In this model a server services some buffer until this buffer empties out. If at that time the other buffer is not empty, the server switches to that buffer. They proved that in this model the above network can become unstable if $m_1 + m_4 > 1$.

Building on this example, Lu and Kumar [LK91] considered a queueing policy that imposes the following priority rules for which buffer the server will serve: buffer B_4 has priority over buffer B_1 at server S_1, and buffer B_2 has priority over buffer B_3 at server S_2. They proved that if $m_1 = m_3 = 0$ and $m_2 = m_4 = \frac{2}{3}$ the system becomes unstable. Dai and Weiss [DW96] proved that the corresponding fluid network is unstable, if and only if $m_2 + m_4 \geq 1$, under the assumptions of the fluid model for the inter-arrival and service times. This is a strong indication that the queueing network is unstable under the same conditions. It remains an interesting open question whether this network is stable under FIFO queueing policy.

Motivated by an example described in [KS90], Rybko and Stolyar [RS92] constructed a new example. They replaced the single customer stream with two types of customers; customers of type 1 that visit servers S_1, S_2, and customers of type 2 that visit servers S_2, S_1. The input stream of customers of both types is a Poisson process of rate 1. The service time of a customer of type i at server S_j follows some distribution with mean m_{ij}. The queueing discipline imposes the same priority rules as in [LK91], i.e., customers of type 1 have priority at server S_2, and customers of type 2 have priority at server S_1. They proved that if $m_{12} = m_{21} > \frac{1}{2}$, the system is unstable. The same result was proven in [DW96] for the corresponding fluid network, under the assumptions of the fluid model for the inter-arrival and service times.

Both examples above are based on priority rules for the servers. Bramson [Bra94a] extended the phenomenon underlying these examples to the case of FIFO queueing policy. Bramson considers a two server network where customers enter the network according to a Poisson process and follow the route $S_1, S_2, \ldots, S_2, S_1$ and then exit the network. The number of visits to the second server is large. The service requirements at each visit along

the route are exponentially distributed random variables with mean c at the first visit to server S_2 and the last visit to server S_1, and mean δ at the first visit to server S_1 and all but the first visit to server S_2. It is demonstrated in [Bra94a] that for c sufficiently close to 1, if the number of visits to server S_2 is sufficiently large and δ is sufficiently small, then the system is unstable under the rate condition. A second example by Bramson [Bra94b] demonstrates that for any arbitrarily small $\rho < 1$, it is possible to have a multiclass network with FIFO queueing policy, where the load factor at each server is less than ρ, but the network is unstable. An interesting corollary derived in [Bra94b] is that decreasing the mean service times within the network may have the effect of making it unstable. Instability of FIFO queueing policy on multiclass networks was also shown by Seidman in [Sei94] for a different model, where arrivals and service times are deterministic. The results of Bramson [Bra94a, Bra94b] and Seidman [Sei94] are some of the most surprising developments in the area of queueing networks.

2.2 Applications to packet routing

A packet routing network is a special case of a queueing system. In packet routing networks the customers are packets that move on a directed graph. The servers are the edges of the graph. The distinctive characteristic of a packet routing network is that all packets require unit service times, whereas the more typical assumption in queueing theory is that the service times are exponentially distributed. This apparently slight difference posed subtle difficulties in adapting queueing theory to packet routing. However, recent advances in the area of generalized Jackson and Kelly networks make it possible to argue directly about stability in packet routing networks, using results in queueing theory.

The problem of packet routing was first considered outside the context of queueing theory ([Lei90],[KL95]). Stamoulis and Tsitsiklis [ST91] were the first to apply results of queueing theory to the special case of layered Markovian networks. In Markovian networks the next edge to be traversed by a packet is a random function of the edge just traversed, and it is independent of the identity of the packet. Packets are assumed to have random destinations. Stamoulis and Tsitsiklis consider such networks under the assumption of Poisson arrivals. They prove that for a layered Markovian network the total number of packets in a network with constant edge traversal times and FIFO queueing policy is statistically dominated

by the total number of packets in a network with a processor sharing queueing policy. Since the latter has a product form solution, queueing theory can yield bounds on the expected queue sizes. They apply this result on layered networks such as the butterfly and the hypercube. Harchol-Balter and Wolfe [HBW95] generalize the result in [ST91] for any Markovian network, showing that the "layered" assumption is not necessary. But they also show that when removing the Markovian assumption there is an example where the delays for FIFO with constant service times are no longer statistically dominated by the delays for processor sharing servers.

Harchol-Balter and Black [HBB94] consider the analysis of packet routing on arrays and toroidal networks. They provide a simple formula for the calculation of the queue sizes in the case that the service times are exponentially distributed and the packets are injected by a Poisson process and have random destinations. Then, they replace exponentially distributed service times with constant, and use experimental results to indicate that the queue sizes in this case are upper bounded by the queue sizes under exponentially distributed service times. They conjecture that this holds for any network. Mitzenmacher [Mit94] proves this conjecture for one bend routing on arrays using the result of Stamoulis and Tsitsiklis. However, the counter-example of Harchol-Balter and Wolfe indicates that it is not possible to obtain bounds for general networks by applying the approach in [ST91].

The question of stability of FIFO queueing policy on packet routing networks was resolved only recently by Bramson [Bra96], using limiting fluid models. Bramson considers networks where the service time of a customer at a server depends on the server and not on the class of the customer that is being serviced (this condition excludes the unstable network described in [Bra94a]). He proves that any queueing network is stable under the FIFO queueing policy, under the assumptions of the fluid model for the inter-arrival and service times [1]. This class of networks includes packet routing networks with constant service times and Poisson arrivals.

A different model was proposed by Cruz [Cru91a, Cru91b] that models the so-called "leaky-bucket" method of flow control proposed for high speed networks. In this model the packets are injected in S sessions; at each session packets that follow a fixed path are injected at a fixed rate (with some burstiness allowed). The total traffic arriving at any

[1] The fluid model assumes that the inter-arrival and service times are independent and identically distributed. Furthermore, the inter-arrival times are assumed to be unbounded and spread out.

edge of the network at any interval of time (s,t) is at most

$$\rho(t-s)+\sigma, \tag{2.1}$$

where $\rho < 1$, and σ is a constant that allows for some burstiness. Cruz [Cru91b] proves the stability of every work-conserving queueing policy on every layered DAG. Tassiulas and Georgiades [TG96] work within this model and prove that every work-conserving queueing policy is stable on the ring.

The Cruz model tries to capture worst case settings and it is in fact a weaker adversarial model. At each one of the S sessions a different type of packets is injected. The paths of packets of different types may share edges. The total traffic at each edge is subject to the condition 2.1. Therefore, for a sufficiently large $w \geq 1$, depending on σ, there exists $0 < r < 1$, such that the amount of traffic injected in any window of w steps is at most $\lfloor rw \rfloor$. Thus, a set of S sessions in the leaky-bucket model corresponds to an adversary of rate less than one in the adversarial model defined in [AAF+96], or in our model. Any stability result in the latter models implies an analogous result in the former model. However, the class of adversaries described by the Cruz model is significantly weaker than the one defined in [AAF+96], or in our model. The Cruz model requires that once some session is set up, it remains permanently active, and it always injects packets at the same rate. The model in [AAF+96] allows the adversary to halt and resume the operation of some session, or to introduce a new session. The rate of some session may vary over time. This broader model is necessary, when modeling connections of limited duration. Thus, stability in the Cruz model may not imply stability in the adversarial model.

Chapter 3

The New Adversarial Model

3.1 Model definition

A queueing network is a graph. Each node consists of a server and a queue of infinite capacity. Customers arrive continuously at the nodes of the network. We assume that each customer has a predefined path: a sequence of servers it must visit, and the service time the customer requires at each server along the path. We assume that every server in the network serves at rate one. When the service of a customer at some server is completed, the customer moves to the next server in zero time. If the server is the last server in its path, the customer exits the system. At any time t at most one customer can be serviced at any server. The conflicts are resolved by the **queueing policy**. In this work we consider only **work-conserving** and **non-preemptive** queueing policies.

Customers are injected into the network by an adversary. We consider time to be continuous. At any time t the adversary injects a set of customers at some nodes of the network. For each server in the path of each customer the adversary specifies the service time that the customer requires at that server. We say that customer c induces workload m for server S at some particular time, if the remaining workload of customer c at server S is m. We say that W amount of workload is queued at server S, if the total workload for S from all the customers queued at, or served by server S is W [1]. We say that the workload (induced) for server S is W, if the total workload for server S induced by all the customers in the system is W. Finally, we say that the total workload in the system is W,

[1] When we talk about customers *queued* at server S we also include the customer that is being served by S.

if the summation of the workload for all servers in $\mathcal{G}$ is W. The rate of an adversary in our work will be specified by a pair (w, μ), where w is any number and $0 < \mu < 1$.

Definition 3.1 *We say that $\mathcal{A}$ is an adversary of rate (w, μ), if for all servers S in the networks, and all continuous intervals I of length w, the customers it injects during I induce at most μw amount of workload for server S.*

Following the definition of Andrews et al. [AAF+96], a system is defined as a triplet $(\mathcal{G}, \mathcal{A}, \mathcal{Q})$, where $\mathcal{G}$ is the underlying graph of the network, $\mathcal{A}$ is the adversary, and $\mathcal{Q}$ is the queueing policy. At any time t:

(i) A set of customers (which may be empty) is injected by the adversary $\mathcal{A}$.

(ii) If the service of some customer c at server S is completed at time t, then the customer advances to the next server in its path in zero time.

(iii) If the queue of server S is non-empty at time t, then the queueing policy $\mathcal{Q}$ selects the next customer to be serviced at server S.

(iv) If the server S in (ii) is the last server in the path of customer c, then c exits the system at time t.

We give the following definition of stability:

Definition 3.2 *A system $(\mathcal{G}, \mathcal{A}, \mathcal{Q})$ is stable if for every initial configuration, there exists some constant C, such that the total workload in the system at all times is bounded by C.*

Following the definitions in [AAF+96] we give the following definitions:

Definition 3.3 *We say that a graph $\mathcal{G}$ is universally stable, if for every $\varepsilon > 0$ and $w \geq 1$ every work-conserving queueing policy is stable against every adversary of rate $(w, 1-\varepsilon)$ on $\mathcal{G}$, for any initial configuration.*

Definition 3.4 *We say that the queueing policy $\mathcal{Q}$ is universally stable, if for every $\varepsilon > 0$ and $w \geq 1$, and every network $\mathcal{G}$ it is stable against every adversary of rate $(w, 1-\varepsilon)$ on $\mathcal{G}$, for any initial configuration.*

In the following lemma we show that the initial configuration is not always important in the definition of universal stability of queueing policies. We prove the lemma for a class of

queueing policies that we call **memoryless** queueing policies, which make scheduling decisions independent of the past history of the customers in the queue. Memoryless queueing policies include policies that make scheduling decisions based on local information at each server (i.e., FIFO, LIFO, MSD, LSD), or based on the future of the customers in the queue (i.e. NTG, FTG, MTTG, LTTG).

Lemma 3.1 *Let $\mathcal{G}$ be a graph, $\mathcal{Q}$ a memoryless queueing policy, and $\mathcal{A}$ an adversary of rate μ. For any system $(\mathcal{G}, \mathcal{A}, \mathcal{Q})$ that starts from a non-empty configuration, there exists a time τ and a system $(\mathcal{G}', \mathcal{A}', \mathcal{Q})$, where A' is an adversary of rate μ and $\mathcal{G}$ is a subgraph of graph $\mathcal{G}'$, such that $(\mathcal{G}', \mathcal{A}', \mathcal{Q})$ starts from an empty configuration and for all $t \geq \tau$ the system $(\mathcal{G}', \mathcal{A}', \mathcal{Q})$ behaves identically with the system $(\mathcal{G}, \mathcal{A}, \mathcal{Q})$.*

Proof: We construct the graph $\mathcal{G}'$ as follows. For each node $v \in \mathcal{G}$, which in the system $(\mathcal{G}, \mathcal{A}, \mathcal{Q})$ begins with k_v customers, we define a set of nodes T_v, which can be thought of as forming a tree rooted at v. T_v has k_v branches, one for each of the k_v customers, which are rooted at node v and do not intersect. Therefore, no customers in T_v meet before they reach their root in v. The service requirements of the customers at the servers of the tree T_v can be chosen arbitrarily as long as they are less than μw. For all v we choose T_v to be sufficiently large that there is an adversary $\mathcal{A}'$ of rate μ that can inject k_v customers into T_v, such that under any work-conserving queueing policy, all customers arrive at their roots in $\mathcal{G}$ at the same time t^*. This is possible since each branch of servers is used exclusively by a single customer. Making the branches sufficiently long and choosing the service times appropriately, we can guarantee that the adversary injects at rate μ, and all customers arrive at their root at the same time, although they are injected at different times. Starting from time t^*, $\mathcal{A}'$ behaves like $\mathcal{A}$, and the system $(\mathcal{G}', \mathcal{A}', \mathcal{Q})$ behaves like $(\mathcal{G}, \mathcal{A}, \mathcal{Q})$. This means that if we observe some node v' in the subgraph $\mathcal{G}$ of $\mathcal{G}'$, and the corresponding node v in $\mathcal{G}$, then, starting from time t^*, the two nodes are indistinguishable. ■

Corollary 3.1 *If a memoryless queueing policy $\mathcal{Q}$ is stable against every adversary $\mathcal{A}$, on every network $\mathcal{G}$, starting from an empty configuration, then $\mathcal{Q}$ is universally stable.*

Corollary 3.1 implies that in order to prove universal stability of some memoryless queueing policy Q, it suffices to prove stability of Q starting from an empty configuration. Furthermore, if some system $(\mathcal{G}, \mathcal{A}, \mathcal{Q})$ is unstable, starting from some non-empty

configuration, then there exists a system $(\mathcal{G}', \mathcal{A}', \mathcal{Q})$ that is unstable starting from an empty configuration, where $\mathcal{A}'$ has the same rate as $\mathcal{A}$.

3.2 Universal stability of queueing policies

Andrews et al. [AAF+96], showed that the queueing policies **Furthest-To-Go (FTG)**, **Longest-In-System (LIS)** and **Shortest-In-System (SIS)** are universally stable in the adversarial packet routing model. The SIS queueing policy gives precedence to the customer most recently injected in the network; the LIS queueing policy gives precedence to the customer injected the earliest in the network; the FTG queueing policy gives precedence to the customer whose distance to its destination is maximal. In case of a tie among customers that have the same priority, we consider an adversary that selects the next customer to be serviced. The adversary may use any priority rule to resolve the tie. We will prove that these queueing policies remain universally stable in the new model. For the following proofs we will assume that the length of the longest path of any customer injected by the adversary is bounded. The proofs of stability follow closely the proofs presented in [AAF+96].

The assumption of bounded path lengths is also stated in [AAF+96]. However, in the adversarial packet routing model the bound on the length of any path is directly implied by the fact that all packets require unit service times at all servers, and no packet can cross an edge more than $(1-\varepsilon)w$ times. This does not hold in our model, since the service requirements of some customer, at some server, may become arbitrarily small, and possibly zero. Thus, the adversary can inject customers with arbitrarily long paths. It remains an interesting question if it is possible to obtain stability results in this case.

Note that the assumption of bounded path lengths still allows the adversary to inject customers that require arbitrarily small service times at some servers. A stronger assumption is that the service requirements of any customer injected by the adversary, at any server, are bounded from below by some constant δ. This condition implies a bound on the length of the longest path of any customer. We later show that within this model, the MTTG queueing policy is universally stable. This model can be slightly generalized by allowing the adversary to inject customers with zero service requirements at some servers. We call this the $(0, \delta)$-model.

3.2.1 SIS is universally stable

Theorem 3.1 *Let $\mathcal{G}$ be a directed network, and $\mathcal{A}$ an adversary of rate $(w, 1-\varepsilon)$, with $\varepsilon > 0$. Then the system $(\mathcal{G}, \mathcal{A}, \mathcal{SIS})$ is stable.*

Proof: Consider some customer c in the system $(\mathcal{G}, \mathcal{A}, \mathcal{SIS})$ at time t waiting in the queue of server S, and suppose that the total workload induced for server S by customers that have priority over c is W. This includes the service requirement of the customer being serviced at the time that c arrives at server S. Since the queueing policy is non-preemptive, the service of this customer must be completed before c becomes eligible for service. Let W' be the first multiple of w greater than W, $W < W' \leq W + w$. We claim that c will start being serviced in the next $\frac{W'}{\varepsilon}$ units of time. Suppose that it is not. Then during each of the next $\frac{W'}{\varepsilon}$ units of time some customer is serviced that has priority over c. But any customer in the system during this time that has priority over c, and requires server S must be one of the customers existing in the system at time t, or injected after time t within the next $\frac{W'}{\varepsilon}$ units of time. Without loss of generality, we assume that $\varepsilon = \frac{1}{n}$ for some natural number n. The total workload induced for server S during this time is at most $\frac{W'}{\varepsilon}(1-\varepsilon)$. Thus the total workload induced for server S by customers that have priority over c during this time is at most $W + \frac{W'}{\varepsilon}(1-\varepsilon) < \frac{W'}{\varepsilon}$, a contradiction. The service requirement of c at server S is at most $w(1-\varepsilon)$; therefore, c will leave server S after at most $\frac{W+w}{\varepsilon} + w(1-\varepsilon)$ time units.

The initial configuration of the system may be non-empty. Denote by L the maximum workload induced for any server by the customers initially in the system. We consider the customers of the initial configuration as being all injected instantaneously at some time before time zero. We define $\beta = w(1-\varepsilon)$, and numbers $W_1, W_2, \ldots$ by the recurrence $W_1 = \max\{\beta, L\} + \beta$, $W_{j+1} = \varepsilon^{-1}(W_j + \beta) + (1-\varepsilon)\beta + 2\beta$. Consider now some customer c whose path visits the servers $S_1, S_2, \ldots, S_{d'}$, where $d' \leq d$ and d is the length of the longest path that the adversary injects in $\mathcal{G}$. Now, by induction on j, we claim that at the time that c arrives at the queue of server S_j, or at any time prior to that, the following holds: for any server S_i in the remaining path of c (including server S_j), the total workload induced for server S_i by customers that have priority over c is at most W_j. This holds for $j = 1$, since for any server S_i, the only customers that want to visit server S_i and have priority over c are the customers injected at the same time as c. The total service time required at server S_i

by these customers is at most L if c is one of the initial customers, or at most $\beta = w(1-\varepsilon)$ if c is one of the customers injected later. Additionally, server S_1 may be servicing some customer at the time that c arrives. The service requirement of this customer at server S_1 is at most $\beta = (1-\varepsilon)w$ units of time. Thus, the total workload induced for server S_i by customers that have priority over c is at most W_1.

Now, suppose that the claim holds for some j. Then by the above argument, c will arrive at server S_{j+1} in at most $\frac{w+W_j}{\varepsilon}+\beta$ units of time after reaching server S_j. The total workload induced for any server of the network by customers injected during this time (which have priority over c) is at most $\lceil(\frac{w+W_j}{\varepsilon}+\beta)/w\rceil(1-\varepsilon)w \leq (1-\varepsilon)\frac{w+W_j}{\varepsilon}+(1-\varepsilon)\beta+\beta$. Server S_{j+1} may be servicing some customer when c arrives. The service time of this customer at server S_{j+1} is at most $\beta = w(1-\varepsilon)$ units of time. Thus, when c arrives at server S_{j+1} the total workload induced for any server by customers that have priority over c is at most

$$W_j + (1-\varepsilon)\frac{w+W_j}{\varepsilon} + (1-\varepsilon)\beta + 2\beta = W_{j+1},$$

and hence the claim holds.

Finally, let m be the number of servers in $\mathcal{G}$. We claim that the total workload in $\mathcal{G}$ is at most $m(W_d + (1-\varepsilon)w)$. For if there was ever $m(W_d+(1-\varepsilon)w)+\delta$ amount of workload in the system, then there would be some server S for which the total workload is $W_d + (1-\varepsilon)w + \delta/m$. Consider now the customer with the lowest priority for server S. The service time required by this customer at server S is at most $(1-\varepsilon)w$ time units. Thus the service time required at server S by customers that have priority over this customer is at least $W_d + \delta/m$ which contradicts the claim of the previous paragraph. ■

3.2.2 LIS is universally stable

Let us denote as **class** l the set of customers injected at time l. Note that this set may be empty if no customers were injected at time l. A class l is said to be **active** at time t if and only if at that time there is some customer in the system of class $l' \leq l$. Note that if ℓ is the smallest active class in the system at time t, then all classes in the interval $[\ell, t]$ are active. We define this interval to be the interval of active classes at time t. We define $r_t = t - \ell$ to be the **range** of active classes at time t.

The initial configuration of the system may be non-empty. Denote by L the maximum

workload induced for any server by the customers initially in the system. We consider the customers of the initial configuration as being all injected instantaneously at some time before time zero. Since these customers are injected before time starts advancing for the system no class is defined for them. The interval of active classes does not include the customers of the initial configuration.

Consider now some customer c, injected at time T_0, whose path visits servers $S_1, S_2, \ldots, S_d$, in this order, where d is the length of the longest path that the adversary injects in $\mathcal{G}$. We use T_i to denote the time that the service for customer c is completed at server S_i [2]. Let t be some time in $[T_0, T_d)$, and define $r = \max_{t \in [T_0, T_d)} r_t$.

Lemma 3.2 $T_d - T_0 < (r + 3w)(1 - \varepsilon^d) + L\frac{1-\varepsilon^d}{1-\varepsilon}$.

Proof: The customer c reaches server S_i at time T_{i-1}. Since c is still in the system, all classes in $[T_0, T_{i-1}]$ are active at time T_{i-1}. The interval of active classes is $[\ell, T_{i-1}]$, where $\ell \leq T_0$. The interval of active classes that can block c in the queue of S_i is $[\ell, T_0]$. From the definition of r the range of that interval is at most $r - (T_{i-1} - T_0)$. The total workload induced for server S_i by customers injected in $[\ell, T_0]$ is at most $\lceil (r + T_0 - T_{i-1})/w \rceil (1-\varepsilon)w < (1 - \varepsilon)(r + T_0 - T_{i-1} + w)$. If there are any customers of the initial configuration still in the system, they have priority over c, and induce at most L amount of workload for server S_i. Thus, the total workload induced for server S_i, by customers that have priority over c is at most $(1 - \varepsilon)(r + T_0 - T_{i-1} + w) + L$. Server S_i may be servicing some customer at the time that customer c arrives. Since the queueing policy is non-preemptive, the service of this customer must be completed before c becomes eligible for service. The service time of any customer at any server is at most $M = (1 - \varepsilon)w$ units of time. Therefore,

$$T_i < T_{i-1} + (1 - \varepsilon)(r + w + T_0 - T_{i-1}) + L + M + M_c,$$

where M_c is the service time required at server S_i by customer c. Therefore,

$$\begin{aligned} T_i &< T_{i-1} + (1 - \varepsilon)(r + w + T_0 - T_{i-1}) + L + 2(1 - \varepsilon)w \\ &= \varepsilon T_{i-1} + (1 - \varepsilon)(r + 3w + T_0) + L. \end{aligned}$$

[2] T_i is well defined since there is some finite time at which the customer c becomes the oldest customer among those queued at S_i, and it is selected to be serviced next.

Thus, solving the recurrence, we obtain

$$\begin{aligned} T_d &< \varepsilon^d T_0 + (1-\varepsilon)(r+3w+T_0)\sum_{i=0}^{d-1}\varepsilon^i + L\sum_{i=0}^{d-1}\varepsilon^i \\ &= T_0 + (r+3w)(1-\varepsilon^d) + L\frac{(1-\varepsilon^d)}{1-\varepsilon} \end{aligned}$$

and the claim follows. ■

Theorem 3.2 *The range of the interval of active classes in the system $(\mathcal{G}, \mathcal{A}, \mathcal{LIS})$ is never more than $\frac{3w}{\varepsilon^d} + \frac{L}{(1-\varepsilon)\varepsilon^d}$, where d is the longest path in $\mathcal{G}$.*

Proof: Let $r = \frac{3w}{\varepsilon^d} + \frac{L}{(1-\varepsilon)\varepsilon^d}$ and assume that at time t, it is the first time that the range of the interval of active classes becomes $r+\delta$ where $\delta > 0$. Hence, at time t there are customers that have been in the system for $r+\delta$ time units, and during the first r time units the range of active classes was no more than r.

However, from the above lemma, any customer that has at most r active classes while in the system (except maybe at time t), is absorbed in at most

$$\begin{aligned} (r+3w)(1-\varepsilon^d) + L\frac{(1-\varepsilon^d)}{1-\varepsilon} &= r - (3w - (3w(1-\varepsilon^d)) - \frac{L - L(1-\varepsilon^d)}{1-\varepsilon} \\ &= r - (3w\varepsilon^d + \frac{L\varepsilon^d}{1-\varepsilon}) \end{aligned}$$

units of time. Since $\varepsilon^d > 0$, $3w\varepsilon^d + \frac{L\varepsilon^d}{1-\varepsilon} > 0$. Therefore, the customer exits the system in less than r units of time, and we reach a contradiction. ■

Corollary 3.2 *Let $\mathcal{G}$ be a directed network, and $\mathcal{A}$ an adversary of rate $(w, 1-\varepsilon)$, with $\varepsilon > 0$. Then the system $(\mathcal{G}, \mathcal{A}, \mathcal{LIS})$ is stable. The workload in the system is never more than $\mathcal{O}(\frac{wm}{\varepsilon^d} + \frac{Lm}{(1-\varepsilon)\varepsilon^d})$, and the maximum time any packet spends in the system is $\mathcal{O}(\frac{w}{\varepsilon^d} + \frac{L}{(1-\varepsilon)\varepsilon^d})$.*

3.2.3 FTG is universally stable

Theorem 3.3 *Let $\mathcal{G}$ be a directed network, and $\mathcal{A}$ an adversary of rate $(w, 1-\varepsilon)$, with $\varepsilon > 0$. Then the system $(\mathcal{G}, \mathcal{A}, \mathcal{FTG})$ is stable.*

Proof: Let m be the number of servers and d be the length of the longest path that the adversary injects in graph $\mathcal{G}$. Let us define $W_i = 0$ for $i > d$, and $W_i = m\sum_{j>i} W_j +$

$2mw(1-\varepsilon)$ for $1 \leq i \leq d$. We prove by a backwards induction on i that the system is stable. We claim that for all $j \geq i$ the workload induced in the system by customers that have *exactly* j servers left to visit (including the server the customers is queued , or processed at) is at most W_j.

This is trivial for $j > d$ since each customer has at most d servers to visit. For some $i \geq 1$, assume that it is true for all $j > i$. Now, consider a particular server S and let $X_i(t)$ be the set of customers that are queued at the queue of server S at time t that still have to visit *at least* i servers. Let t be the current time, and let t' be the most recent time preceding time t, at which $X_i(t')$ became non-empty (i.e., $X_i(t') > 0$ and $X_i(t'-\delta) = 0$, for any $\delta > 0$). Any customer in $X_i(t)$ must either have had at least $i+1$ servers to visit at time t' (since it must have been at least one server away from S), or else it was injected after time t'. From the induction hypothesis the total workload induced for any server by customers that had exactly $j > i$ servers to visit at time t' is at most W_j. If at time t' some customer was being serviced at server S, then since the queueing policy is non-preemptive the service of this customer must have been completed before the server S started servicing customers from $X_i(t')$. The service requirement of any customer at any server is $\alpha \leq (1-\varepsilon)w$ units of time, so customers from $X_i(t')$ become eligible for service at time $t' + \alpha$. If $\alpha > t - t'$, then no customer from $X_i(t')$ has been serviced. Otherwise, from the definition of the queueing policy at every time $t'' \in [t' + \alpha, t]$ some customer from $X_i(t'')$ was serviced at server S. Consider some arbitrary server S' in $\mathcal{G}$. We define $T_i(t)$ to be the workload induced by the customers in $X_i(t)$, for S'.

$$T_i(t) \leq \sum_{j>i} W_j + (1-\varepsilon)w \left\lceil \frac{t-t'}{w} \right\rceil - (t-t') + \alpha,$$

where $\sum_{j>i} W_j + (1-\varepsilon)w$ is an upper bound on the amount of workload induced for server S' by customers in $X_i(t)$ that had $j > i$ servers to visit at time t', and $(1-\varepsilon)w\lceil \frac{t-t'}{w} \rceil$ is the amount of workload induced for server S' by customers in $X_i(t)$ that were injected during the interval $[t', t]$. Since the queueing policy is work-conserving, $t - t'$ amount of workload was serviced during this interval; $\min\{\alpha, t-t'\} \leq \alpha$ time units of this workload is induced by a customer that does not belong to $X_i(t)$. Since $\alpha \leq (1-\varepsilon)w$,

$$T_i(t) \leq \sum_{j>i} W_j + 2(1-\varepsilon)w - \varepsilon(t-t').$$

The above inequality has two consequences. First, the workload induced by customers that have to visit exactly i servers, for any server in $\mathcal{G}$, is always at most $m\sum_{j>i} W_j + 2m(1-\varepsilon)w = W_i$ and so the inductive step holds. Second, $t - t'$ cannot be greater than $\frac{1}{\varepsilon}(\sum_{j>i} W_j + 2(1-\varepsilon)w)$. Hence this expression gives the maximum amount of time that a customer with i servers to visit can remain in a queue. Therefore under FTG the total workload in the system is bounded by $\sum_{j\geq 1} W_j$, and the maximum amount of time that any customer spends in the system is bounded by $\frac{W_1}{\varepsilon}$. ■

3.3 Instability of the ring

Andrews et al. [AAF+96], proved that the ring is universally stable in the adversarial packet routing model. Any packet routing network can be easily turned into a queueing network, simply by replacing the edges with servers. The resulting network is a unidirectional ring of servers; if $S_0, \ldots, S_{n-1}$ are the nodes of the ring, then a customer that completes its service at server S_i can only move to server $S_{(i+1) \bmod n}$, or exit the network. The result in [AAF+96] implies that if the service times required by the customers are the same at any server of the network, then the ring of servers is universally stable. A similar result was proven in [DW96], using limiting fluid models, under the assumptions of the fluid model for the inter-arrival and service times. Stability of the ring under any queueing policy was also proven in [TG96] for the "leaky bucket" model defined by Cruz in [Cru91a, Cru91b]. We show that this does not hold if the customers can require different service times at different servers. There are simple queueing policies, such as **Nearest-To-Go (NTG)**, **Most-Service-Demand (MSD)**, **Least-Time-To-Go (LTTG)** and **First-In-First-Out (FIFO)**, for which there is an adversary of rate $(w, 1-\varepsilon)$ that causes them to become unstable on the ring. The NTG queueing policy gives precedence to the customer whose distance from its destination is minimal. The MSD queueing policy gives precedence to the customer that requires the maximum service time at a given server among those queued at that server. The LTTG queueing policy gives precedence to the customer whose remaining time is minimal. The remaining service time of some customer c is the summation of the service times required by c at all servers in its path that c has not visited yet (including the one c is queued at).

It may seem surprising that a system where workload is serviced faster than it is induced at each server can be made unstable. The adversary forces instability by causing some

servers to remain idle. Workload is induced for these servers, but it is blocked at other servers. Therefore, the total service capacity of the system becomes less than the load induced by the customers, which causes the system to become unstable.

3.3.1 Instability of NTG, LTTG and MSD on the ring

We will prove that the NTG queueing policy is unstable on the two node ring. The example we describe is a simple generalization of the examples in [RS92] and [LK91].

Let $\mathcal{G}$ be the two node ring. Let S_1 and S_2 be the two servers of the ring. Rybko and Stolyar [RS92] consider two types of customers: customers of type 1 that visit servers S_1, S_2 and customers of type 2 that visit servers S_2, S_1. The queueing policy gives priority to customers of type 1 at server S_2 and to customers of type 2 at server S_1. Note that these rules define the NTG queueing policy. It is easy to prove that the same example is unstable in the adversarial context, under the same conditions on the service times. The analysis uses a similar technique to that described by Lu and Kumar in [LK91].

Theorem 3.4 *Let $\mu > \frac{1}{2}$. There exists an adversary $\mathcal{A}$ of rate $(1, \mu)$ such that the system $(\mathcal{G}, \mathcal{A}, \mathcal{NTG})$ is unstable, starting from some non-empty configuration.*

Proof: The adversary $\mathcal{A}$ injects two types of customers: customers of type 1 that visit servers S_1, S_2, with service requirements m_1 and m_2, and customers of type 2 that visit servers S_2, S_1, with service requirements m_3 and m_4. For simplicity we assume that $m_2 = m_4 = m$, and $m_1 = m_3 = \varepsilon$. Furthermore, we assume that $\varepsilon < m$.

We break the construction of $\mathcal{A}$ into phases. Our induction hypothesis will be as follows: at the beginning of phase j, there will be at least t_j customers of type i queued at server S_i, where $t_j = ca^j$, for some constants c and $a > 1$. Depending on whether j is odd or even, $i = 1$ or 2.

To start out, however, we need s_0 customers of type 1 queued at server S_1. Thus, setting $c = s_0$, the induction hypothesis for phase 0 is certainly met. For a general phase j (assume that J is odd) we will show that if at the beginning of phase j the queue of server S_1 contains a set of $s = a^j s_0$ customers of type 1, then at the start of phase $j+1$, there will be at least $a^{j+1} s_0$ customers of type 2 in the queue of server S_2.

The adversary starts injecting customers immediately after the first customer of type 1 has reached server S_2. For each of the next $\lfloor \frac{m}{1-m} s \rfloor$ time units we inject exactly one

customer of each type. Since $\varepsilon < m$, there is always at least one customer of type 1 at server S_2 which blocks the customers of type 2. The total workload induced for server S_2 during this time by customers of type 1 is $m\lfloor \frac{m}{1-m} s \rfloor + ms$. Since server S_2 serves customers at rate 1, by the end of this period there exists at most one customer of type 1 in the system, queued at server S_2, with residual service time

$$m \left\lfloor \frac{m}{1-m} s \right\rfloor + ms - \left\lfloor \frac{m}{1-m} \right\rfloor s = ms - (1-m) \left\lfloor \frac{m}{1-m} s \right\rfloor = (1-m)r,$$

where $0 \leq r < 1$ is the decimal part of $\frac{m}{1-m}$. The adversary waits for this customer to complete its service. This ends phase j.

The total number of customers of type 2 queued at server S_2, at the end of phase j is $\lfloor \frac{m}{1-m} s \rfloor > \frac{m}{1-m} s - 1$. For some arbitrarily small $\theta > 0$, we can select s_0 to be sufficiently large so that

$$\frac{m}{1-m} s_0 - 1 \geq (1-\theta) \frac{m}{1-m} s_0.$$

By induction hypothesis, $s > s_0$. Therefore, $\frac{m}{1-m} s - 1 \geq (1-\theta) \frac{m}{1-m} s$. We set $a = (1-\theta) \frac{m}{1-m}$. In order for the induction hypothesis to be met for phase $j+1$ we need

$$(1-\theta) \frac{m}{1-m} > 1.$$

Since $m < 1$, the above inequality is satisfied when $m > \frac{1}{2} + \frac{\theta}{2}$. Therefore, $\mu = m + \varepsilon > \frac{1}{2} + \frac{\theta}{2} + \varepsilon$, where θ and ε can be chosen arbitrarily small. The adversary $\mathcal{A}$ has rate $(1, \mu)$ and the system $(\mathcal{G}, \mathcal{A}, \mathcal{NTG})$ is unstable. ■

For simplicity of the proof, and in order to achieve the lowest possible bound on the rate μ we assumed that $m_2 = m_4$, and $m_1 = m_3$. We may consider a more general case, where the service times are not the same. It is not hard to show that in this case the system is unstable when $m_2 + m_4 > 1$, and $m_1 < m_2$ and $m_3 < m_4$.

The queueing policy NTG gives priority to customers of type 1 at server 2, and to customers of type 2 at server 1. Note that the customers of type 1 require more service time at server 2 than customers of type 2, and that customers of type 2 require more service time at server 1 than customers of type 1. Furthermore, the remaining time of customers of type 1 queued at server 2 is less than the remaining time of customers of type 2 queued at the same server; the remaining time of customers of type 2 queued at server 1 is less than

that of customers of type 1. Therefore, in this network, and for this adversary, the NTG queuing policy is identical with LTTG and MSD queueing policies.

Corollary 3.3 *Let $\mu > \frac{1}{2}$. There exists an adversary $\mathcal{A}$ of rate $(1, \mu)$ such that the system $(\mathcal{G}, \mathcal{A}, \mathcal{LTTG})$ is unstable, starting from a non-empty configuration.*

Corollary 3.4 *Let $\mu > \frac{1}{2}$. There exists an adversary $\mathcal{A}$ of rate $(1, \mu)$ such that the system $(\mathcal{G}, \mathcal{A}, \mathcal{MSD})$ is unstable, starting from a non-empty configuration.*

3.3.2 Instability of FIFO on the ring

We define the graph $\mathcal{G}_n$ to be the unidirectional $2n$-node ring. We denote by $S_0, \ldots, S_{2n-1}$ the servers of the ring. For the proof of the following theorem we assume that the adversary may inject customers that require zero service time at some server. It is easy to prove that the theorem still holds when we replace 0 by an arbitrarily small ε.

Theorem 3.5 *Let $\mu > \frac{1}{2}$. There is an n, and an adversary $\mathcal{A}$ of rate $(1, \mu)$ such that the system $(\mathcal{G}_n, \mathcal{A}, \mathcal{FIFO})$ is unstable, starting from some non-empty configuration.*

Proof: Consider $\theta > 0$ such that $\mu > \frac{1}{2} + \theta$, and let $\lambda = (1 - \theta)\mu$. One can check that $\frac{\lambda}{1-\lambda} > 1$. For reasons that will become clear later, we select n so that $\frac{\lambda(1-\lambda^n)}{1-\lambda} > 1$. We break the construction of $\mathcal{A}$ into phases. Our inductive hypothesis will be as follows: at the beginning of phase j, there will be at least t_j customers in the queue of server S_{in} that want to visit servers $S_{in}, S_{in+1} \ldots, S_{(i+1)n-1}$ and require service time μ at each server, where $t_j = ca^j$, for some constants c and $a > 1$. Depending on whether j is even or odd, $i = 0$ or 1.

To start out, however, we need s_0 customers queued at S_0. For reasons that will become clear later we choose s_0 such that $\lfloor \lambda^n \mu s_0 \rfloor \geq (1 - \theta)\lambda^n \mu s_0$. Furthermore, s_0 is sufficiently large so that for any $A \geq \lambda^n \mu s_0$,

$$\lfloor A \rfloor \geq (1 - \theta) A. \tag{3.1}$$

These s_0 customers want to visit servers $S_0, \ldots, S_{n-1}$, and require service time μ at each server. Thus, setting $c = s_0$, the induction hypothesis for phase 0 is certainly met. Let j be a general phase, and without loss of generality assume that j is even. Assume that at the beginning of phase j the queue of server S_0 consists of a set of $s \geq s_0$ customers that

want to visit servers $S_0, \ldots, S_{n-1}$, with required service times μ at each server. We will show that at the start of phase $j+1$, there will be at least $\frac{\lambda(1-\lambda^n)}{1-\lambda}s$ customers in the queue of server S_n, that want to visit servers $S_n, \ldots, S_{2n-1}$, with required service time μ at each server, We can therefore let $a = \frac{\lambda(1-\lambda^n)}{1-\lambda} > 1$.

We break the construction of phase j into n sub-phases, $0, 1, \ldots, n-1$. The duration of sub-phase k is W_k time units, where W_k is the amount of workload queued at server S_k at the beginning of sub-phase k. During this time, for $\lfloor W_k \rfloor$ steps, the adversary injects exactly one customer that wants to visit servers $S_k, \ldots, S_{n-1}, S_n, \ldots, S_{2n-1}$ with required service time 0 at servers $S_k, \ldots, S_{n-1}$ and μ at servers $S_n, \ldots, S_{2n-1}$. Furthermore, if $k < n-1$, the adversary also injects exactly one customer that wants to visit servers $S_{k+1}, \ldots, S_{n-1}$ with required service time μ at each server.

For simplicity we introduce the following definitions. We say that a customer c wants to complete a full circle, if for some $0 \leq k \leq n-1$, c is queued at S_k and wants to visit servers $S_k, \ldots, S_{n-1}, S_n, \ldots, S_{2n-1}$, with service requirements 0 at servers $S_k, \ldots, S_{n-1}$ and μ at $S_n, \ldots, S_{2n-1}$. We say that a customer c wants to complete a half circle, if for some $0 \leq k \leq n-1$, c is queued at S_k and wants to visit servers $S_k, \ldots, S_{n-1}$, with service requirements μ at all servers.

We now give an informal description of the behavior of the system. At any point in time there may exist only two types of customers in the system: customers that want to complete a full circle, and customers that want to complete a half circle. We will show that half-circle customers always block full-circle customers. Consider some sub-phase k, where k is not the first or the last phase. We will show that at the beginning of sub-phase k, there exists a set of half-circle customers queued at server S_k, and a set of full-circle customers queued at server S_{k-1}. There exist no customers at servers $S_0, \ldots, S_{k-2}$. There may exist some half-circle customers at servers $S_{k+1}, \ldots, S_{n-1}$, left over from the previous sub-phase. There exist no customers at servers $S_n, \ldots, S_{2n-1}$. Full-circle customers move to server S_k. Furthermore, the adversary injects new full-circle customers at server S_k. Since the queueing policy is FIFO, all these full-circle customers are blocked by the half-circle customers which were there first. These half-circle customers drain out of server S_k during sub-phase k. The adversary injects more half-circle customers at server S_{k+1} which partially replace the half-circle customers that are absorbed. Sub-phase k ends when all the half circle customers at server S_k move to server S_{k+1}. Note that during sub-phase k

full-circle customers are blocked at server S_k, and therefore no customers go beyond server S_{n-1}. Also no customers are injected for servers $S_0, \ldots, S_{k-1}$, or $S_n, \ldots, S_{2n-1}$. Thus, all the activity in the network is concentrated at servers $S_k, \ldots, S_{n-1}$.

We will show by induction that at the end of each sub-phase the number of full-circle customers in the system increases. We will prove that the full-circle customers accumulated at the end of the last sub-phase satisfy the inductive hypothesis for phase $j+1$.

Denote by $N = \mu s$ the workload queued at server S_0 at the beginning of phase j. We will prove by induction that for any sub-phase k, $0 \leq k \leq n-1$ the following hold:

1. At the beginning of sub-phase k there exists at least $\lambda^k N$ amount of workload queued at server S_k, induced by customers that want to complete a half circle.

2. At the end of sub-phase k, there exist at least $(1-\theta)(1+\lambda+\ldots+\lambda^k)N$ customers queued at server S_k that want to complete a full circle [3].

At the beginning of sub-phase 0, there exists N amount of workload queued at server S_0, induced by customers that want to complete a half circle. During the N time units of sub-phase 0 the adversary injects $\lfloor N \rfloor$ customers that want to complete a full circle which are queued at server S_0. Note that these customers are blocked at server S_0 since they have lower priority. By 3.1, $\lfloor N \rfloor \geq (1-\theta)N$. Thus, at the end of sub-phase 0 there exist at least $(1-\theta)N$ customers, so the induction hypothesis is satisfied for sub-phase 0.

Consider now a general sub-phase k, $0 < k \leq n-1$. By induction hypothesis, at the beginning of sub-phase $k-1$ there exists $W_k \geq \lambda^{k-1}N$ amount of workload queued at server S_{k-1} that is induced by customers that want to complete a half circle. During sub-phase $k-1$ the workload queued at server S_{k-1}, moves to server S_k. The adversary injects $\lfloor W_k \rfloor \geq \lfloor \lambda^{k-1}N \rfloor$. customers at server S_k that want to complete a half circle. By 3.1, $\lfloor W_k \rfloor \geq (1-\theta)\lambda^{k-1}N$. These customers induce at least $\mu(1-\theta)\lambda^{k-1}N \geq \lambda^k N$ amount of workload for server S_k. Since the queueing policy is work-conserving the amount of workload serviced during phase $k-1$ is W_k. Thus, at the beginning of phase k there exists at least $\lambda^k N + W_k - W_k = \lambda^k N$ amount of workload queued at server S_k, induced by customers that want to complete a half circle. The first part of the induction hypothesis is satisfied for sub-phase k.

[3] We adopt the convention that the end of sub-phase k is the point in time when the workload initially at server S_k has been serviced, but before the customers with zero service time at server S_k have moved to the next server.

By the induction hypothesis, at the end of sub-phase $k-1$, there exist at least $(1-\theta)(1+\lambda+\ldots+\lambda^{k-1})N$ customers queued at server S_{k-1} that want to complete a full circle. During sub-phase k these customers move to server S_k. Furthermore, the adversary injects at least $(1-\theta)\lambda^k N$ customers that want to complete a full circle, which are queued at server S_k. All these customers are blocked at server S_k by customers that want to complete a half circle, which have higher priority. Thus, at the end of sub-phase k there are at least $(1-\theta)(1+\lambda+\ldots+\lambda^k)N$ customers queued at server S_k that want to complete a full circle. Therefore, the second part of the induction hypothesis is also satisfied for sub-phase k, so the induction hypothesis is met for sub-phase k. This concludes the induction on the sub-phases of phase j.

Therefore, at the end of phase j there exist at least $(1+\lambda+\ldots+\lambda^{n-1})N$ customers queued at server S_{n-1} that want to visit servers $S_{n-1}, S_n, \ldots, S_{2n-1}$, with service requirements 0 at server S_{n-1} and μ at all other servers. We consider customers with zero service time to be moving instantaneously, so at the beginning of phase $j+1$ all customers that want to visit server S_n are in the queue of server S_n. The number of customers queued at server S_n is at least

$$s' = (1-\theta)(1+\lambda+\ldots+\lambda^k)N = (1-\theta)\frac{1-\lambda^n}{1-\lambda}\mu s = \frac{\lambda(1-\lambda^n)}{1-\lambda}s.$$

Thus, the induction hypothesis is met for phase $j+1$. ■

3.4 The MTTG queueing policy

The **Most-Time-To-Go (MTTG)** queueing policy gives precedence to the customer whose remaining service time in the system is maximal. The remaining service time of some customer c is the summation of the service time required by c at all servers in its path that c has not visited yet (including the one c is queued at). We will prove that MTTG is stable against a class of restricted adversaries, where the adversary is restricted to injecting customers with service requirements bounded from below. That is, there exists some $\delta > 0$, such that for any customer c, the service time required by c at any server in its path is at least δ. The proof follows closely the proof of stability of FTG presented in [AAF+96] and the one presented in section 3.2.3. If we allow the adversary to inject customers that require arbitrarily small service time at some servers, then we will show that there exists

a network $\mathcal{G}$ on which the queueing policy MTTG is unstable. The instability example for MTTG is similar to the one for the NTG queueing policy. In order for the adversary to force instability, it injects customers that require service times that are a decreasing function of the time.

An interesting question is how to resolve ties between customers of different types queued at the same server that have the same remaining service time. Ideally, we would prefer an adversary to select the next customer to be serviced. The adversary may use any priority rule to resolve the tie. In the case that the service times are bounded form below, the proof of stability of MTTG that we present considers worst case settings, and it is not affected by the decisions of the adversary that resolves the ties. In the $(0, \delta)$-model [4], however, it is not hard to see that the instability example described for NTG in section 3.3.1, works also for the MTTG queueing policy. Specifically, if we set $m_1 = m_3 = 0$ then all customers that meet at any server have the same remaining service time. Selecting the NTG priority rule to resolve the ties causes the system to become unstable. It remains an interesting question if it is possible to force instability in the $(0, \delta)$-model when there are no ties.

3.4.1 Stability of MTTG

Theorem 3.6 *Let $\mathcal{G}$ be a directed network, and $\mathcal{A}$ an adversary of rate $(w, 1-\varepsilon)$, with $\varepsilon > 0$. The service time required at any server, by any customer injected by $\mathcal{A}$, is at least δ. Then the system $(\mathcal{G}, \mathcal{A}, \mathcal{MTTG})$ is stable.*

Proof: Denote by m the number of servers in the network. The total service time required by any customer is at most $D = mw(1-\varepsilon)$. Let $d = \lceil \frac{D}{\delta} \rceil$. Let us define $W_i = 0$ for $i > d$, and $W_i = m \sum_{j>i} W_j + 2mw(1-\varepsilon)$ for $1 \leq i \leq d$. For $j \geq 1$, define Δ_j to be the interval $[j\delta, (j+1)\delta)$. We prove by a backwards induction that the system is stable. We claim that for all $j \geq 1$ the workload induced in the system by customers with remaining service time in Δ_j is at most W_j.

This is trivial for $j > d$ since each customer requires at most D total service time. For some $i \geq 1$, assume that it is true for all $j > i$. Now, consider a particular server S and let $X_i(t)$ be the set of customers that are queued at the queue of server S at time

[4] In the $(0, \delta)$-model the service requirement of some customer at some server may be zero, but the non-zero service requirements of the customers are bounded from below by some constant δ.

t with remaining service time at least $i\delta$ [5]. Let t be the current time, and let t' be the most recent time preceding time t, at which $X_i(t')$ became non-empty (i.e., $X_i(t') > 0$ and $X_i(t' - \gamma) = 0$, for any $\gamma > 0$). Any customer in $X_i(t)$ must either have been injected after time t', or else it must have been at least one server away from S at time t', and therefore it must have had remaining service time at least $(i+1)\delta$. From the induction hypothesis, for all $j > i$ the total workload induced for any server by customers with remaining service time in Δ_i at time t' is at most W_j. If at time t' some customer was being serviced at server S, then since the queueing policy is non-preemptive the service of this customer must have been completed before the server S started servicing customers from $X_i(t')$. The service requirement of any customer at any server is $\alpha \leq (1-\varepsilon)w$ units of time, so customers from $X_i(t')$ become eligible for service at time $t' + \alpha$. If $\alpha > t - t'$, then no customer from $X_i(t')$ has been serviced. Otherwise, from the definition of the queueing policy at every time $t'' \in [t' + \alpha, t]$ some customer from $X_i(t'')$ was serviced at server S. We define $T_i(t)$ to be the workload induced by the customers in $X_i(t)$, for any server in $\mathcal{G}$. Following the argument for the proof of stability of FTG queueing policy,

$$\begin{aligned} T_i(t) &\leq \sum_{j>i} W_j + (1-\varepsilon)w \left\lceil \frac{t-t'}{w} \right\rceil - (t-t') + \alpha \\ &\leq \sum_{j>i} W_j + 2(1-\varepsilon)w - \varepsilon(t-t'). \end{aligned}$$

The above inequality has two consequences. First, the workload induced by customers that have remaining service time in Δ_i is always at most $m\sum_{j>i} W_j + 2m(1-\varepsilon)w = W_i$, and so the inductive step holds. Second, $t - t'$ cannot be greater than $\frac{1}{\varepsilon}(\sum_{j>i} W_j + 2(1-\varepsilon)w)$. Hence, this expression gives the maximum amount of time that a customer with remaining service time in Δ_i can remain in a queue. Therefore, under MTTG the total workload in the system is bounded by $\sum_{j\geq 1} W_j$, and the maximum amount of time that any customer spends in the system is bounded by $\frac{W_1}{\varepsilon}$. ■

[5] This is the main difference between the proofs of stability of FTG and MTTG. In the former case, the induction is on the number of edges remaining in the path of some customer. In the latter case, the induction is on the remaining time, which is discetized into intervals of size δ.

3.4.2 Instability of MTTG

We define $\mathcal{G}$ to be a network with four servers. We denote by S_0, S_1 and V_0, V_1 the servers of the network. The servers S_0 and S_1 form a 2-node ring, and the servers V_0 and V_1 are attached to S_0 and S_1 respectively. For simplicity, in the following theorem we assume that the adversary may inject customers that require zero service time at some server. We will remove this assumption later on.

Theorem 3.7 *Let $\mu > \frac{1}{2}$. There is an adversary $\mathcal{A}$ of rate $(1, \mu)$ such that the system $(\mathcal{G}, \mathcal{A}, \mathcal{MTTG})$ is unstable, starting from a non-empty configuration.*

Proof: The adversary $\mathcal{A}$ injects two types of customers: customers of type 0 that visit servers S_1, S_0, V_0, and customers of type 1 that visit servers S_0, S_1, V_1. We break the construction of $\mathcal{A}$ into phases. Our inductive hypothesis is as follows: at the beginning of phase j, there exist at least t_j customers of type i queued at server S_i, where $t_j = ca^{j-1}$ for some constants c and $a > 1$. These customers want to visit servers S_i, V_i, with service requirements μ and $\frac{\mu}{j}$ respectively. Depending on whether j is even or odd, $i = 0$ or 1. For the sake of simplicity, in the following we assume that $\mu = \frac{2}{3}$. We will later show that the theorem can be generalized for μ arbitrarily close to $\frac{1}{2}$.

For convenience, we consider the first phase of the system to be phase 1. To start out we need s_1 customers of type 1 at the queue of server S_1, for $s_1 = 1$. The customers require service time μ at both servers S_1 and V_1. Thus, if we set $c = s_1$ the induction hypothesis is met for phase 1. Consider now a general phase j and assume that j is even. Assume that at the beginning of phase j there exist $s = (\frac{\mu}{1-\mu})^{j-1} s_1$ customers of type 0 queued at server S_0, that require service time μ and $\frac{\mu}{j}$ at servers S_0 and V_0. We will show that at the beginning of phase $j+1$ there exist $(\frac{\mu}{1-\mu})^j s_1$ customers of type 1 queued at server S_1, that require service time μ and $\frac{\mu}{j+1}$ at servers S_1 and V_1. We set $a = \frac{\mu}{1-\mu} = 2$.

The duration of phase j is $\frac{\mu}{1-\mu} s$ time units. For each of the next $\frac{\mu}{1-\mu} s$ time units the adversary injects exactly one customer of each type. Customers of type 0 visit servers S_1, S_0, V_0, with service requirements $0, \mu$ and $\frac{\mu}{j}$. Customers of type 1 visit servers S_0, S_1, V_1 with service requirements $0, \mu$ and $\frac{\mu}{j+1}$. Customers of type 0 have priority over customers of type 1 at server S_0. At the end of phase j there exist no customers of type 0 in the system, and there exist $\frac{\mu}{1-\mu} s$ customers of type 1 queued at server S_0. We consider customers with zero service requirement to be moving instantaneously. So, at the beginning of phase $j+1$

there exist $\frac{\mu}{1-\mu}s = (\frac{\mu}{1-\mu})^j s_1$ customers of type 1 at the queue of server S_1. Thus, the induction hypothesis is met for phase $j+1$.

It is not hard to prove that the theorem can be generalized for any μ arbitrarily close to $\frac{1}{2}$. Let s be the number of customers queued at server S_0 at the beginning of phase j. The duration of phase j is again $\frac{\mu}{1-\mu}s$ time units, but the adversary injects $\lfloor \frac{\mu}{1-\mu}s \rfloor$ customers. At the end of phase j the adversary waits for the last customer of type 0 to complete its service, before entering phase $j+1$. Consider some $\theta > 0$, such that $\mu > \frac{1}{2} + \theta$. We select s_1 such that $\lfloor (\frac{\mu}{1-\mu})s_1 \rfloor \geq (1-\theta)(\frac{\mu}{1-\mu})s_1$. This holds if and only if $\theta(\frac{\mu}{1-\mu})s_1 > 1$. By induction hypothesis $s > s_1$; therefore the number of customers of type 1 queued at server S_1 at the beginning of phase $j+1$ is $\lfloor (\frac{\mu}{1-\mu})s \rfloor \geq (1-\theta)(\frac{\mu}{1-\mu})s$. The number of customers increases if $(1-\theta)(\frac{\mu}{1-\mu}) > 1$, which holds since $\mu > \frac{1}{2} + \theta$. ■

During the phase j the service requirements at servers V_0 and V_1 are $\frac{\mu}{j}$ and $\frac{\mu}{j+1}$. These service times can be replaced by any strictly decreasing function $f(j)$, such that $0 < f(j) < \mu$, for all j. Furthermore, the zero service requirements can be replaced by some appropriately chosen strictly decreasing function $g(j)$, such that $0 < g(j) < 1 - \mu$. In this case, during the phase j, the adversary injects customers of type 0 that require service times $g(j), \mu, f(j)$, and customers of type 1 that require service times $g(j+1), \mu, f(j+1)$. In order for customers of type 0 to have priority over customers of type 1 at server S_0, we require that

$$\mu + f(j) > g(j+1) + \mu + f(j+1) \Rightarrow$$

$$f(j) - f(j+1) > g(j+1).$$

Selecting the functions f and g appropriately we can ensure that the instability example works for the case that all customers have non-zero service requirements. Since the customers no longer move instantaneously between servers, the adversary should follow the sequence of injections described in the instability example for NTG presented in section 3.3.1. The rate of the adversary is $(1, \mu + g(1))$. Selecting $g(1)$ to be arbitrarily small, we can make the rate of the adversary to be arbitrarily close to $\frac{1}{2}$. Possible values for the functions f and g are $f(j) = \frac{\mu}{j}$ and $g(j) = \varepsilon^j$, where $0 < \varepsilon < 1 - \mu$.

For simplicity we chose to present the instability example for MTTG on the graph $\mathcal{G}$ described above. It is not difficult to show that MTTG can be made unstable on the

2-node ring. Let S_0, S_1 be the servers of the ring. The adversary injects two types of customers: customers of type 0 that visit servers S_0, S_1, and customers of type 1 that visit servers S_1, S_0. We break the construction of the adversary into phases. In every phase, the adversary injects customers of both types. The service requirements of the customers are $g(j), \mu+f(j)$ and $g(j+1), \mu+f(j+1)$ respectively, for g and f previously defined. Following the analysis of the previous example, and that of the instability example for NTG, it is easy to prove that if we initialize the system with s_1 customers of type 1 queued at server S_1, then at the beginning of phase j there exists $(\frac{\mu}{1-\mu})^j s_1$ customers of type i queued at server S_i, where $i = j \bmod 2$. The rate of the adversary is $\mu + f(1) + g(1)$, which can be driven arbitrarily close to $\frac{1}{2}$ by selecting $g(1)$ and $f(1)$ arbitrarily small.

Chapter 4

Instability of NTG for constant rate injections

In this chapter we will show that the queueing policy Nearest-To-Go (NTG) is unstable on some network $\mathcal{N}$, when workload is injected at arbitrarily small rate. The NTG queueing policy gives precedence to the packet whose distance to its destination is minimal. We will prove that for any $\varepsilon > 0$, there exists a network $\mathcal{N}$ such that NTG is unstable for injection rate less than ε, in a model significantly weaker than the one described in chapter 3. In contrast to the examples described in [Bra94a, Bra94b] for FIFO queueing policy, the customers visit each server at most once, and require unit service times at each server. The instability example was first described, and informally proven in [BKSW96], and it was motivated from the study of packet routing networks. We give a formal proof in this chapter. We also show that the network of servers $\mathcal{N}$ can be transformed into a packet routing network that is also unstable for the same rate. For the rest of the discussion we refer to the customers as packets.

4.1 Model description

For the instability example we want to describe, it is advantageous to have a model that it is as weak as possible. We consider time to advance in discrete time steps. We assume that packets require one time step of service time at every server in their path, and that servers serve one packet at each time step. The packets are injected at constant rate. The initial configuration of the system is determined by the adversary. The injection process

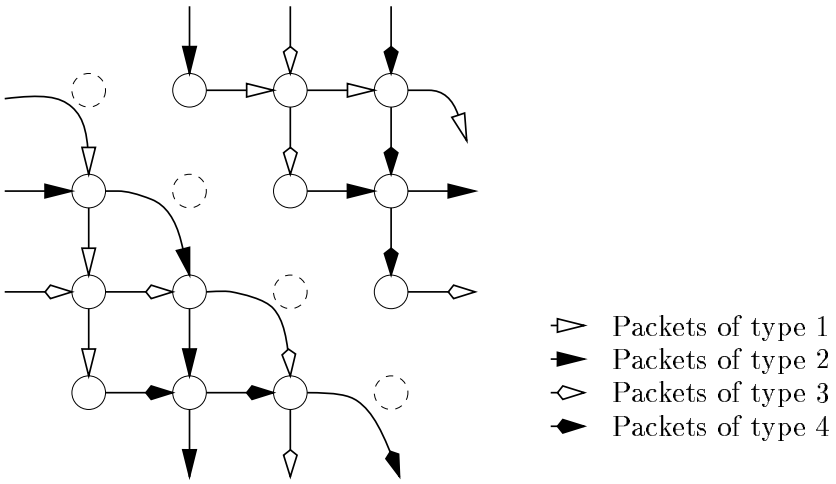

Figure 4.1: The 4×4 network $\mathcal{N}$.

can be thought of as a collection of paths $\mathcal{P}$ on the network $\mathcal{N}$. Each different path in $\mathcal{P}$ defines a different **type** of packet. Let K be the number of different types of packets in the system and $1, \ldots, K$ a numbering of the different types. We assume that all packet types are injected in the network at the same rate r, and all packets are injected in the network simultaneously. Customers are injected at constant rate, that is, one packet of type k is injected every $1/r$ steps.

The network $\mathcal{N}$ we consider is the $n \times n$ torus. The servers are marked as (i, j), where i denotes the row and j the column, and $0 \leq i, j \leq n-1$. From now on all additions and subtractions are considered to be modulo n. There are n types of packets. The packet of type i is generated at node $(i, i+1)$ and follows the sequence

$$(i, i+1), (i, i+2), (i, i+3), \ldots, (i, i-1), (i+1, i), (i+2, i), \ldots, (i-1, i),$$

and then is absorbed. Note that the nodes of the form (i, i) are never used and therefore can be considered as non-existent. An example of the 4×4 network $\mathcal{N}$ is shown in the Figure 4.1.

The arrival rate at any server is defined as the summation of the rates of all types of packets that use that server. The requirement on $\mathcal{P}$ is the following: for any server S in $\mathcal{N}$, the arrival rate at that server should be strictly less than the service rate of S. All customers are injected in the network at rate r. Since every server is used by exactly two

types of packets the arrival rate at every server is $2r$. We assume that the service rate is 1 for all servers. Therefore, we require that $r < \frac{1}{2}$. We denote by $\mathcal{P}(r)$ this collection of paths.

Each of the servers (i, j) is used by exactly two types of packets: packets of type i that move horizontally, and packets of type j that move vertically. Under the queueing policy NTG, the packets of type j always have priority over the packets of type i. We will prove that the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$, is unstable for $r > \frac{2}{n}$, where n is the number of rows or columns of $\mathcal{N}$. For the rest of the thesis we assume that n is even.

We now give some intuition of the proof. A packet that moves vertically has priority over all packets of other types, and it intersects with all other types of packets. Therefore, if we manage to create a continuous stream of packets that move vertically along a column, we will cause all other packets to be blocked at this column. We call this a wall. A wall in the system causes a great number of servers to remain idle, and the number of packets queued at the column of the wall to increase. When the stream empties out we want a new wall to be created in the next column. In order for this transition to be done orderly, it is useful to consider a system with two walls, evenly spaced. We prove that the two walls move together, and each time they move one column over, the total number of packets in the system has increased.

4.2 Definitions

We now introduce the notion of a **wall**.

Definition 4.1 *A wall at column i of system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ is defined by:*

- *a queue of packets of type i at node $(i, i-1)$, which is called the* **feeding queue** *of the wall,*
- *one packet of type i at every node of column i,*
- *queues of packets of type j, $j = 0 \ldots n-1$, $j \neq i$ queued up at node (j, i) that want to cross column i.*

It should be intuitively obvious why this is called a wall. The packets that move down the column i have priority over all packets that want to cross column i, thereby blocking them

at column i. If there is some node in column i that has no packet of type i, then we say that a **hole** is created in the wall. For emphasis, a wall with no holes is said to be **solid**.

We intend to build a system that most of the time it will have two walls $\frac{n}{2}$ columns apart. Let i and $j = i + \frac{n}{2}$ be the columns of the two walls.

The set of nodes between the wall at column j and the wall at column i is called the **span** of the wall at column i. For some row k, $k \in \{0 \ldots n-1\}$ we define the span, $span_i(k)$ to be all the nodes on row k, from the generating node $(k+1, k)$ to the node (k, i) at column i, which are between the columns j and i. Formally:

$$span_i(k) = \{(k, k+1), \ldots, (k, i)\} \cap \{(k, j+1), \ldots, (k, i)\}.$$

Similarly, for the wall at column j, we define

$$span_j(k) = \{(k, k+1), \ldots, (k, j)\} \cap \{(k, i+1), \ldots, (k, j)\}.$$

Denote by $queue(i, j)$ the packets queued up at node (i, j), and by $queue_k(i, j)$ the packets of type k queued up at node (i, j). Obviously, $queue(i, j) = queue_i(i, j) \cup queue_j(i, j)$. For notational convenience, we define $queue(i, i) = \emptyset$. We define the effective queue Q_k of the wall at column i, for the row k, $k \in \{0 \ldots n-1\}$, as

$$Q_k = \bigcup_{v \in span_i(k)} queue_k(v).$$

The effective queues of the wall at column j are defined similarly.

We define the **configuration** of the wall at column i as the vector $(Q_0, Q_1, \ldots, Q_{i-1}, Q_i, Q_{i+1}, \ldots, Q_{n-1})$, where Q_i is the feeding queue, and Q_k is the effective queue at column k, where $k = 0 \ldots n-1$, $k \neq i$. The queue Q_{i+1} is called the **secondary queue** of the wall. We call the queues $Q_i, \ldots, Q_{i+\frac{n}{2}-1}$ the **top queues** of the wall, and the queues $Q_{i+\frac{n}{2}}, \ldots, Q_{i-1}$ the **bottom queues** of the wall [1].

The packets that do not belong to any effective queue are called **runaway** packets. Runaway packets, are the packets that have crossed both column i and j, and are free to

[1] The choice of names "top" and "bottom" may seem awkward to the reader. We consider the wall starting from the feeding queue and going downwards. Thus, the "top" queues are the first $\frac{n}{2}$ queues, while the "bottom" queues are the next $\frac{n}{2}$ queues. This choice will be justified later, where we show that the top queues are the next to become the feeding queues of the wall.

head to their destination. As soon as they start moving vertically they have priority over the packets moving horizontally, and they may block packets that belong to the effective feeding queues.

Now, we introduce the notion of contiguousness of the effective queues. An effective queue Q_k for wall i is **contiguous** if,

$$queue_k(k,l) = 0 \Rightarrow queue_k(k,l') = 0,\ \forall (k,l') \in span_i(k) \backslash \{(k,l+1),\ldots,(k,i)\}.$$

Intuitively, this means that the packets are queued up at consecutive nodes and there are no gaps between the queues.

We now introduce the notion of symmetry.

Definition 4.2 *The system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ is in a symmetric state if, $\forall k,l,m \in \{0,\ldots,n-1\},\ l \neq m$,*

$$|queue_k(l,m)| = |queue_{k+\frac{n}{2}}(l+\frac{n}{2}, m+\frac{n}{2})|.$$

The nodes (l,m) and $(l+\frac{n}{2}, m+\frac{n}{2})$ are called **corresponding nodes**, and the types of packets k and $k+\frac{n}{2}$ are called **corresponding types**. Definition 4.2 says that the corresponding nodes have the same number of packets of corresponding types. Note that if n is even, then the relation defined by correspondence is symmetric: if (x,y) is the corresponding node of node (l,m), then (l,m) is the corresponding node of (x,y). The same holds for the types of packets. Corresponding nodes service corresponding types of packets. Packets of corresponding types move along corresponding nodes, that is, for all i, the k-th node in the path of a packet of type i is the corresponding node of the k-th node in the path of a packet of type $i+\frac{n}{2}$.

Consider now a system with two walls at columns i and j. Let $(Q_0, Q_1, \ldots, Q_{n-1})$ be the configuration of the wall at column i, and $(Z_0, Z_1, , \ldots, Z_{n-1})$ the configuration of the wall at column j. If the system is symmetric, then Definition 4.2 implies that the two walls are $\frac{n}{2}$ columns apart, i.e., $j = i + \frac{n}{2}$. Otherwise, the corresponding nodes of nodes at columns i and j would have different queue sizes. Furthermore, it holds that, $\forall k \in \{0,\ldots,n-1\}$,

- $\forall (k,l) \in span_i(k),\ (k+\frac{n}{2}, l+\frac{n}{2}) \in span_{i+\frac{n}{2}}(k+\frac{n}{2})$,
- and $|Q_k| = |Z_{k+\frac{n}{2}}|$.

The queues Q_k and $Z_{k+\frac{n}{2}}$ are called **corresponding effective queues**.

We now introduce the notion of a **phase**. We will show that the evolution of the system in time can be divided into such phases.

Definition 4.3 *The system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ enters a phase at column i if the following conditions are satisfied:*

(C0) The system is symmetric.

(C1) There exist two walls at columns i and $i + \frac{n}{2}$.

(C2) The feeding queues of the two walls are contiguous.

(C3) There exist no runaway packets.

The phase terminates when the feeding queues empty out and there exist no packets of type i and $i + \frac{n}{2}$ at the nodes of columns i and $i + \frac{n}{2}$ respectively.

A phase breaks into two periods: the **solid period** and the **transition period**. The solid period is the period of time that the two walls remain solid, while the transition period is the period of time that the two walls collapse and the last packets of the walls drain out of the system.

We now give a sketch of the proof of instability. We first show that symmetry is preserved throughout time; if we initialize the system to be symmetric, it will remain symmetric. We then prove that the walls remain solid, as long as there are packets in the feeding queues. When the feeding queues empty out the solid period terminates and the system enters the transition period. In Lemma 4.3 we show that if a set of conditions *(C4)–(C5)* is satisfied at the beginning of the phase, then at the end of the transition period the system will enter a new phase, with the walls one column over. We then prove that in this new phase the total number of packets has increased. Finally, we show that if the system is initialized appropriately, then at the beginning of any phase the conditions *(C4)–(C5)* are always satisfied.

4.3 System properties

Lemma 4.1 *If there exists some time step t_0 at which the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ is symmetric, then it will remain symmetric for every t, $t > t_0$.*

Proof: We will use induction on time. The system is symmetric at time step t_0. Assume now that the system is symmetric at time t. There exist two main types of events that can happen in time step t in the system: (1) a new packet is injected, (2) a packet is served and advanced to the next node (a packet that is absorbed can be considered as being advanced to a null node). New packets that are injected in the system are injected simultaneously at corresponding nodes. Therefore, the symmetry is preserved. Consider now a node (i, j) and the corresponding node $(i + \frac{n}{2}, j + \frac{n}{2})$. There are two types of packets queued up at node (i, j): packets of type i that move horizontally, and packets of type j that move vertically. Packets of type j have priority over packets of type i. The corresponding node $(i + \frac{n}{2}, j + \frac{n}{2})$ has packets of corresponding types $i + \frac{n}{2}$ and $j + \frac{n}{2}$, with type $j + \frac{n}{2}$ having higher priority. By definition of symmetry, corresponding nodes have the same number of packets of corresponding types. Since the queueing policy used at all nodes is the same, the packets selected to be served next will be of corresponding types.

Packets of corresponding types move along corresponding nodes; therefore the packets served at nodes (i, j) and $(i+\frac{n}{2}, j+\frac{n}{2})$ are advanced to corresponding nodes. Furthermore, since n is assumed to be even, the corresponding node of $(i + \frac{n}{2}, j + \frac{n}{2})$ is (i, j). Therefore, when a packet is advanced to the next node, the relations among the queues of corresponding nodes remain the same; thus symmetry is preserved.

Since all events that happen in time-step t preserve symmetry, at the beginning of time step $t + 1$ the system will be symmetric. ■

Lemma 4.1 guarantees that in a symmetric system with two walls the corresponding effective queues always have the same size and change in the same way. Furthermore, it guarantees that if the system is initialized to be symmetric, then condition *(C0)* will always be satisfied. For the rest of this section we assume that the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ is initialized to be symmetric.

In the following lemma we consider the behavior of the system during the solid period of the phase.

Lemma 4.2 *Consider a system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ that enters a phase at column i. Let N be the size of the feeding queues of the two walls. The conditions (C1)–(C3) are satisfied for the next N steps.*

Proof: Let $(Q_0, Q_1, \ldots, Q_{n-1})$ and $(Z_0, Z_1, \ldots, Z_{n-1})$ be the configurations of the walls at columns i and $i + \frac{n}{2}$ respectively. We consider only the wall at column i, but since the system is symmetric the same also hold for the wall at column $i + \frac{n}{2}$.

Let T be the time that the system enters the phase. We claim by induction that for any time step $t \in [T, T + N]$ the conditions *(C1)–(C3)* are satisfied. This holds at time T. Now, assume that it holds at the beginning of time step $t \leq T + N$. We will prove that the conditions will be satisfied at the beginning of time step $t + 1$.

First note that, since the effective feeding queue has initially N packets, and $t \leq T + N$, there exists at least one packet at the feeding queue, at the beginning of time step t. There are two ways a hole can be created in the wall in time step t: if the feeding queue Q_i is non–contiguous, or if the packets of Q_i are blocked at node $(i, i-1)$ by some runaway packet of type $i-1$. From the induction hypothesis, the queue Q_i is contiguous at the beginning of time step t, and there exist no runaway packets. Thus, at the beginning of time step $t + 1$, the condition *(C1)* is satisfied.

There are two ways that the feeding queue can become non–contiguous in time step t: if a packet is added to the feeding queue, or if some runaway packet blocks the packets of the feeding queue at some node in $span_i(i)$. From the induction hypothesis there exist no runaway packets at time t. Furthermore, since the two walls are solid at the beginning of time step t, no packets of the queue Z_i cross column $i + \frac{n}{2}$ to enter queue Q_i. Consider now some newly injected packet p of type j. The packet p follows the sequence

$$(j, j+1), (j, j+2), (j, j+3), \ldots, (j, j-1), (j+1, j), (j+2, j), \ldots, (j-1, j).$$

If $j \in \{i \ldots i + \frac{n}{2} - 1\}$, then the packet p will reach column $i + \frac{n}{2}$ at node $(j, i + \frac{n}{2})$ before it reaches column i, and it will be queued at the effective queue Z_j. If $j \in \{i + \frac{n}{2} \ldots i - 1\}$, then the packet p will reach column i at node (j, i) before it reaches column $i + \frac{n}{2}$, and it will be queued at the effective queue Q_j. Therefore, the queues fed are the bottom queues of the walls. No new packets are added to the top queues, including the feeding queues. Therefore the queue Q_i is contiguous at the beginning of time step $t + 1$ and the condition *(C2)* is satisfied.

Since the walls are solid in time step t, no packets cross columns i and $i + \frac{n}{2}$. From the above argument, packets injected in time step t are queued at the effective queues

$Q_{i+\frac{n}{2}}, \ldots, Q_{i-1}$ and $Z_i, \ldots, Z_{i+\frac{n}{2}-1}$. Therefore, no runaway packets are created in time step t. The condition *(C3)* is satisfied at the beginning of time step $t+1$.

The walls will remain solid, as long as there are packets in the feeding queues. Since no new packets are added to the feeding queues, they will empty out after N steps. ■

The duration of the solid period is equal to the size of the feeding queues. When the feeding queues empty out the system enters the transition period. In the following lemma we provide a set of conditions *(C4)* and *(C5)* that must be satisfied so that, at the end of the phase, the system enters a new phase at column $i+1$.

Lemma 4.3 *Consider a system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ that enters a phase at column i at time t. Let N be the size of the feeding queues of the two walls at time t. The phase is terminated at time $t' = t + N + n - 1$. If the following conditions (C4) and (C5) are satisfied at time t, then at the end of time step $t'+1$ the system enters a new phase at column $i+1$:*

(C4) The secondary queues have size at least $n-1$.

(C5) The secondary queues are contiguous.

Proof: First note that if the conditions *(C4)* and *(C5)* are satisfied at the beginning of the phase, then they will also be satisfied at the beginning of the transition period, since no packets enter or leave the secondary queues during the solid period. Let $T = t + N$ be the time the solid period ends, i.e., the time that the feeding queues empty out. Let $(Q_0, \ldots, Q_{n-1})$ and $(Z_0, \ldots, Z_{n-1})$ be the configurations of the walls at column i and $i + \frac{n}{2}$ respectively. We examine the behavior of the wall at column i. Since the system is symmetric, the same hold for the wall at column $i + \frac{n}{2}$.

At time T there exist $n-1$ packets of type i at column i, one at each node. Denote by p the packet at node $(i+1, i)$, which will be the last of these packets to leave the system. At time $T+1$ the packet p leaves the node $(i+1, i)$, and frees the packets in the queue Q_{i+1}. At the next time step the packets of type $i+1$ start moving vertically along column $i+1$. At time $T+2$ the first of these packets moves to node $(i+2, i+1)$. Denote this packet by q. There are two streams of packets: one of packets of type i that drain out of column i, and one of packets of type $i+1$ that take over column $i+1$. Packet p is at the tail of the first stream, while packet q is at the head of the second stream. Packets p and q move in

parallel. At time $T+k$, $k \geq 2$, packet q leaves node $(i+k, i)$ and frees the packets of the queue Q_{i+k}. At the same time step packet q moves to node $(i+k, i+1)$ and blocks the packets of the queue Q_{i+k}. At time step $T+n-1$ the packet p exits the system and the phase is terminated. At time step $T+n$ the packet q reaches the node $(i, i+1)$.

From *(C4)*, the queue Q_{i+1} has at least $n-1$ packets. If the stream of these packets is contiguous then at time $T+n$ a wall is formed at column $i+1$. From *(C5)*, the queue Q_{i+1} is contiguous. The only way that the stream of packets of type $i+1$ can be interrupted is if some packet blocks the packets of queue Q_{i+1} at some node in $span_i(i+1)$.

For the following we think of the effective queues Q_k, $0 \leq k \leq n-1$, as "moving" from column i to column $i+1$. Therefore, although the wall at column i collapses, it still makes sense to talk about the queue Q_k. The effective queue Q_k should be thought of as a collection of packets that is identified by row k rather than a queue that is located at some node.

We now claim inductively that the following holds: at the beginning of any time step $t \in [T, T+n]$, no packet has crossed both columns $i+1$ and $i+\frac{n}{2}+1$. In Lemma 4.2 we showed that no runaway packets are created during the solid period; therefore the claim holds at time T. Assume now that the claim is true for any time $t' < t$, where $T \leq t \leq T+n$. From the induction hypothesis at the beginning of time step t there exists no packet that has crossed both columns $i+1$ and $i+\frac{n}{2}+1$, that could block the packets of Q_{i+1}. Thus the stream of packets of type $i+1$ remains contiguous in time step t. From the description of the transition period, for every row k, $k \neq i+1$, there exists a packet that blocks either node (k, i) or node $(k, i+1)$, with only exception the row i. Therefore, no packet of queues $Q_{i+2}, \ldots, Q_{i-1}$ crosses column $i+1$. Symmetrically, no packet of queues $Z_{i+\frac{n}{2}+2}, \ldots, Z_{i+\frac{n}{2}-1}$ crosses column $i+\frac{n}{2}+1$. New packets injected in the interval $[T, T+n]$, of type $i, \ldots, i+\frac{n}{2}-1$ are queued at queues $Z_i, \ldots, Z_{i+\frac{n}{2}-1}$, while new packets of type $i+\frac{n}{2}, \ldots, i-1$ are queued at queues $Q_{i+\frac{n}{2}}, \ldots, Q_{i-1}$. Therefore, the packets of type i that cross column $i+1$ are queued at queue $Z_{i+\frac{n}{2}}$ and do not cross column $i+\frac{n}{2}+1$. Symmetrically, the packets of type $i+\frac{n}{2}$ that cross column $i+\frac{n}{2}+1$ do not cross column $i+1$. Thus, no packets cross both columns $i+1$ and $i+\frac{n}{2}+1$ in time step t.

Since the streams of packets of type $i+1$ and $i+\frac{n}{2}+1$ are not blocked during the transition period, at the end of time step $T+n$, there exist two walls at columns $i+1$ and $i+\frac{n}{2}+1$, and the condition *(C1)* is satisfied. The feeding queues of the two walls are the

queues Q_{i+1} and $Z_{i+\frac{n}{2}+1}$. The two queues are initially contiguous. During the transition period no new packets are added to the queues Q_{i+1} and $Z_{i+\frac{n}{2}+1}$, and no runaway packets block their packets; therefore,they remain contiguous at the end of time step $T+n$, and the condition *(C2)* is satisfied. From the above argument, there exist no runaway packets at the end of time step $T+n$. The conditions *(C0)–(C3)* are satisfied, so at the end of time step $T+n$ the system enters a phase at column $i+1$. ■

Consider now the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ that has entered a phase at column i. Let $(Q_0, Q_1, \ldots, Q_{n-1})$ and $(Z_0, Z_1, \ldots, Z_{n-1})$ be the configurations of the walls at columns i and $i+\frac{n}{2}$ respectively. We make the following observations, which are directly implied by Lemmas 4.2 and 4.3.

Observation 1 *If N is the size of the feeding queue, the duration of the phase is $N+n-1$ steps.*

Observation 2 *During the phase only the bottom queues, $Q_{i+\frac{n}{2}}, \ldots, Q_{i-1}$ and $Z_i, \ldots, Z_{i+\frac{n}{2}-1}$, are fed; no packets are added to the top queues $Q_i, \ldots, Q_{\frac{n}{2}-1}$ and $Z_{\frac{n}{2}}, \ldots, Z_{i-1}$.*

Observation 3 *During the phase exactly two types of packets are absorbed at each time step.*

The last observation is the heart of the proof of the following theorem.

Theorem 4.1 *If $r > \frac{2}{n}$, at the end of a phase the total number of packets in the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ has increased.*

Proof: Consider a system that enters a phase at column i. Let N denote the size of the feeding queues. The total number of packets of type either i or $i+\frac{n}{2}$ is $M = N+n-1$. Denote by O the total number of packets of type other than i and $i+\frac{n}{2}$. The total number of packets in the system T, at the beginning of the phase, is $T = 2M+O$.

The duration of the phase is exactly M steps. During these M steps the total number of packets injected in the network is nrM. During the phase exactly two types of packets are absorbed, so the total number of packets absorbed is $2M$. The total number of packets T' at the end of the phase is $T' = T + nrM - 2M = O + nrM$. In order for $T' > T$, we need that $nrM > 2M$, that is, $r > \frac{2}{n}$. So, if the rate of injection is greater than $\frac{2}{n}$, then the total number of packets at the end of the phase increases. ■

4.4 The Proof of Instability

In the previous section we examined some phase in isolation. We showed that if the conditions *(C4)–(C5)* are satisfied at the beginning of the phase, then at the end of the phase conditions *(C0)–(C3)* are satisfied, and the system enters a new phase. However, we cannot guarantee that conditions *(C4)–(C5)* will be satisfied at the beginning of the new phase. We will now prove that if we initialize the system appropriately then the conditions *(C4)–(C5)* are always satisfied at the beginning of every phase. Thus, once the system enters phase 0, it goes through successive phases.

We initialize the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ with two walls, one at node 0 and one at node $\frac{n}{2}$. All queues are contiguous. All queues have size at least $n-1$, except for the feeding queues whose size is chosen arbitrarily, but it is the same for both queues. There exist no runaway packets. The system is initialized so that it is symmetric. We choose the rate of injection r to be $\frac{3}{n}$, where n is the size of the torus.

Let $(Q_0, \ldots, Q_{n-1})$ and $(Z_0, \ldots, Z_{n-1})$ be the configurations of the two walls. In the following theorem we will prove that the system goes through a sequence of phases. At each phase there exist two walls which collapse at the end of the phase and two new walls are created one column over. Instead of thinking that two new walls are formed at each phase, we can think of the initial two walls as "moving" from column to column. Together with the wall "move" the effective queues associated with this wall. An effective queue can be thought of as a collection of packets identified by the the row index of the queue, and not by the node at which the queue is located. Thus, it makes sense to consider the evolution of an effective queue through different phases of the system.

For the proof of the next theorem we assume that all column indices are modulo n.

Theorem 4.2 *At time $t = 0$, the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ enters phase 0 at column 0. For every $m \geq 0$, at the end of phase m the system enters phase $m+1$ at column $m+1$.*

Proof: The system is initialized so that the conditions *(C0)–(C5)* are satisfied. Therefore, at time $t = 0$ the system enters phase 0. At the end of phase 0, the system enters phase 1 at column 1.

Now assume that, for any $j < k$, at the end of phase j at column j the system enters phase $j+1$ at column $j+1$. We will prove that at the beginning of phase k the conditions *(C4)–(C5)* are satisfied; therefore, at the end of phase k the system will enter phase $k+1$

at column $k+1$. The secondary queues at phase k are the queues Q_{k+1} and $Z_{k+\frac{n}{2}+1}$. For the rest of the proof we consider only the queue Q_{k+1}; since the system is symmetric the same also hold for the queue $Z_{k+1+\frac{n}{2}}$.

We first prove that the queue Q_{k+1} has size at least $n-1$. If $k < n-1$, then the secondary queue is one of the queues that were initialized with size at least $n-1$. During any phase the only packets consumed are the packets that create the walls, so the size of Q_{k+1} either remains the same, or increases. Therefore, the size of Q_{k+1} will be at least $n-1$.

If $k \geq n-1$, then at phase $k-n+1$ the queue Q_{k+1} was the feeding queue. At the beginning of phase $k-n+2$ the queue Q_{k+1} is empty. From phase $k-n+2$ to phase $k-\frac{n}{2}+1$ the queue Q_{k+1} is one of the bottom queues; therefore, from Observation 2 it is fed. From phase $k+\frac{n}{2}+2$, when queue $Q_{k-\frac{n}{2}+1}$ becomes the feeding queue, to phase k the queue Q_{k+1} is one of the top queues; therefore, from Observation 2 it is not fed. Thus, the queue Q_{k+1} is fed for exactly $\frac{n}{2}$ phases. The duration of a phase is $N+n-1$ steps, where N is the size of the feeding queues; therefore, the duration of any phase is at least $n-1$ steps. Therefore, at least $r(n-1)$ packets are queued at Q_{k+1} during each phase that it is fed. Thus, the total size of Q_{k+1} is:

$$|Q_{k+1}| \geq \frac{n}{2} r(n-1).$$

Choosing $r > \frac{2}{n}$ guarantees that the size of Q_{k+1} is at least $n-1$. For $r = \frac{3}{n}$, $|Q_{k+1}| \geq \frac{3}{2}(n-1)$.

We will now show that the queue Q_{k+1} is contiguous. From Lemmas 4.2 and 4.3, no runwaway packets are created during a phase that could block the packets of queue Q_{k+1}. Therefore, the only way that the queue Q_{k+1} can become non-contiguous is if a new packet is added to it. Denote by p the phase $k-\frac{n}{2}+2$, the first phase that Q_{k+1} is not fed. If at the beginning of phase p the queue Q_{k+1} is contiguous, then it will remain contiguous, since Q_{k+1} is not fed and the queueing policy is work conserving. Suppose now that at the beginning of the phase p the queue Q_{k+1} is not contiguous. We will show that there is enough time in phase p for the queue Q_{k+1} to become contiguous. The span of any effective queue is at most $\frac{n}{2}$ nodes. Therefore there can be at most $\frac{n}{2}-1$ empty nodes between two non-empty nodes of an effective queue. Phase p has duration at least $n-1$ steps. Since the

queueing policy is work conserving, at the end of phase p , Q_{k+1} will be contiguous, and it will remain contiguous until phase k.

At the beginning of phase k the conditions *(C4)–(C5)* are satisfied; therefore, at the end of phase k the system enters phase $k+1$ at column $k+1$. ■

We have therefore proved our goal. The following theorem is directly implied from Theorems 4.2 and 4.1.

Theorem 4.3 *The system* $(\mathcal{N}, \mathcal{P}(\frac{3}{n}), \mathcal{NTG})$ *is unstable, starting from a non–empty configuration.*

The instability example we described has very specific initial conditions. We can prove that it is possible to achieve instability starting from a less restrictive initial state of the system. We initialize our system with two queues of packets of type 0 and $\frac{n}{2}$ at nodes $(0, n-1)$ and $(\frac{n}{2}, \frac{n}{2}-1)$ respectively. Both queues have size N, where N is chosen appropriately. We can prove that after some time the system evolves into a state identical to the initial state of the instability example we described. Using Theorem 4.2 we can prove that the system is unstable.

Furthermore, simulation results indicate that it is possible to reach instability starting from a less "artificial" initial condition. Consider node $(i, i-1)$ and the corresponding node $(i+\frac{n}{2}, i-1+\frac{n}{2})$. Assume that these servers are out of service for a sufficiently long period of time. When the two servers resume service the queues build up at these nodes cause the system to become unstable.

4.5 Instability of packet routing networks

The network $\mathcal{N}$ previously described is a network of servers. We are interested in proving instability in the special case of packet routing. We will prove that there exists a packet routing network that is unstable under the same rate conditions. A packet routing network is a directed graph $\mathcal{G}$. The service points of the network are the edges of the graph. The packets follow paths of edges on $\mathcal{G}$.

In the following, we will present a simple algorithm that takes as input network $\mathcal{N}$ and produces a packet routing network $\mathcal{G}$. However, the packet routing network $\mathcal{G}$ is not directly equivalent to the network $\mathcal{N}$. That is, at some time t, there can be some packet p that

is not at the same stage of its route in both $\mathcal{G}$ and $\mathcal{N}$ networks. Thus, we cannot argue directly for the instability of network $\mathcal{G}$. Instead, we transform the network $\mathcal{G}$ back into a network of servers $\mathcal{N}'$ that is unstable if and only if $\mathcal{G}$ is stable. The network $\mathcal{N}'$ can be shown to be unstable for injection rate $r = \frac{3}{n}$.

The algorithm for producing the packet routing network $\mathcal{G}$ is the following:

> For every server k in $\mathcal{N}$, put an edge $e_k = (u_k, v_k)$ in $\mathcal{G}$. If there exists a packet in the network of servers that moves from server k to server m, put an edge $e_{km} = (v_k, u_m)$ in $\mathcal{G}$. We call edges e_k "server" edges, and edges e_{km} "connecting edges". A packet of type i, that follows the path
>
> $$(i, i+1), (i, i+2), \ldots, (i, i-1), (i+1, i), \ldots, (i-1, i)$$
>
> in network $\mathcal{N}$, will follow the path
>
> $$e_{(i,i+1)}e_{(i,i+1)(i,i+2)}e_{(i,i+2)}, \ldots, e_{(i,i-1)}e_{(i,i-1)(i+1,i)}, e_{(i+1,i)}, \ldots e_{(i-1,i)}$$
>
> in network $\mathcal{G}$.

The resulting system $(\mathcal{G}, \mathcal{P}_{\mathcal{G}}(r), \mathcal{NTG})$ is not directly equivalent to the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$. In this network all edges have processing time 1, including the connecting edges. In the network of servers the packets move between the servers in zero time. We will show that it is possible to transform the system $(\mathcal{G}, \mathcal{P}_{\mathcal{G}}(r), \mathcal{NTG})$ back into a network of servers that is unstable, under the same rate conditions.

The transformation of the packet routing network into a network of servers is straightforward. For every edge of the network, create a server, and then connect the servers. The resulting network is an augmented version of the torus $\mathcal{N}$ previously described (Figure 4.2). For every node (i, j) of $\mathcal{N}$ there exist two new nodes (i, j') and (i', j), which represent the connecting edges $e_{(i,j)(i,j+1)}$ and $e_{(i,j)(i+1,j)}$ respectively in graph $\mathcal{G}$. For the special case of nodes $(i, i-1)$ we add a node (i', i) which represents the connecting edge $e_{(i,i-1)(i+1,i)}$. No node $([i-1]', i)$ is introduced for the terminal nodes $(i-1, i)$. We denote this new network by $\mathcal{N}'$.

There still exist n types of packet in the system. A packet of type i in this network

follows the path

$$(i, i+1), (i, [i+1]'), (i, i+2), (i, [i+2]'), \ldots, (i, i-1), (i', i), (i+1, i), ([i+1]', i), \ldots, (i-1, i).$$

Note that the newly introduced nodes are used by exactly one type of packets; therefore no more than one packet is ever queued at these nodes. All packets are injected simultaneously at the same rate r. We denote by $\mathcal{P}'$ this collection of paths. We have defined a new system $(\mathcal{N}', \mathcal{P}'(r), \mathcal{NTG})$. It is easy to see that $(\mathcal{G}, \mathcal{P}_{\mathcal{G}}(r), \mathcal{NTG})$ is unstable if and only if the system $(\mathcal{N}', \mathcal{P}'(r), \mathcal{NTG})$ is unstable.

We consider the network $\mathcal{N}'$ as having n rows and n columns. Row i includes all nodes $(i, \cdot)$, and column j all the nodes $(\cdot, j)$. An example of the 4×4 network $\mathcal{N}'$ is shown in Figure 4.2. The shaded nodes are the newly introduced nodes.

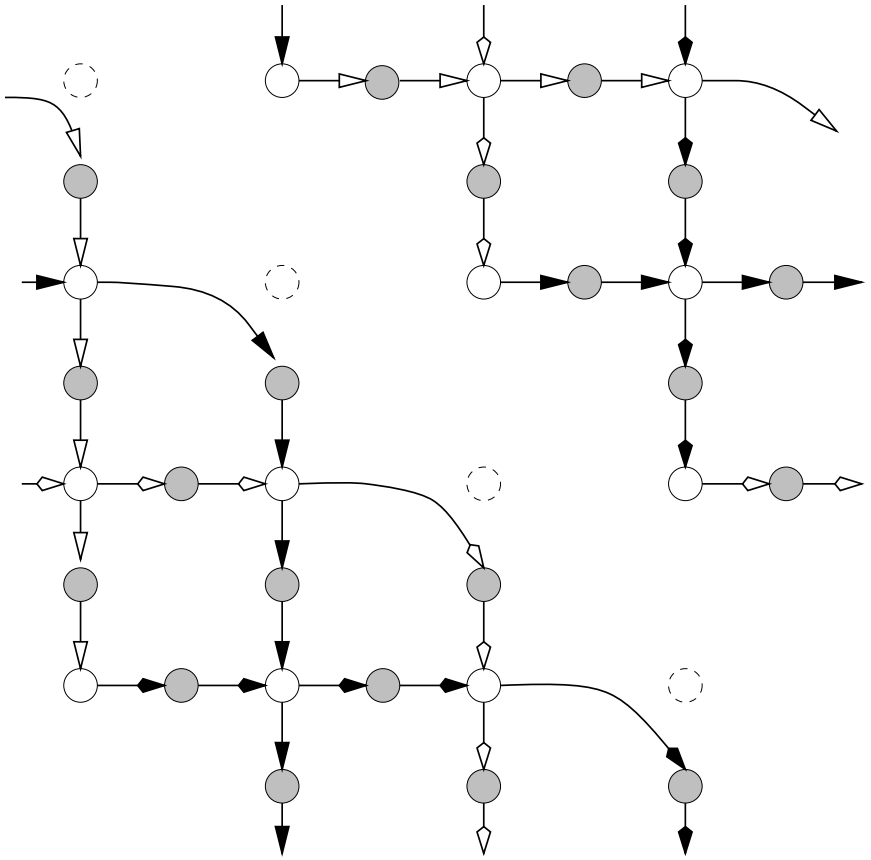

Figure 4.2: The 4×4 network $\mathcal{N}'$

.

The proof of instability for the system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ can be easily modified to work

for system $(\mathcal{N}', \mathcal{P}'(r), \mathcal{NTG})$. The properties of system $(\mathcal{N}, \mathcal{P}(r), \mathcal{NTG})$ can be generalized for the case of the system $(\mathcal{N}', \mathcal{P}'(r), \mathcal{NTG})$. Using these properties, we can show that the system $(\mathcal{N}', \mathcal{P}'(r), \mathcal{NTG})$, when initialized appropriately, evolves in phases. At the end of each phase the number of packets in the system increases. Thus, we reach the following theorem.

Theorem 4.4 *The system $(\mathcal{N}', \mathcal{P}'(\frac{3}{n}), \mathcal{NTG})$ is unstable, starting from a non–empty configuration; therefore, the system $(\mathcal{G}, \mathcal{P}_{\mathcal{G}}(\frac{3}{n}), \mathcal{NTG})$ is unstable starting from a non–empty configuration.*

Chapter 5

Conclusions

In this work we extended the adversarial model described in [BKR+96] and [AAF+96] to include general queueing networks. In the new adversarial model we considered four new queueing policies: Most-Time-To-Go (MTTG), Least-Time-To-Go (LTTG), Most-Service-Demand (MSD), and Least-Service-Demand (LSD). We proved that in this new model the ring is no longer universally stable; the queueing policies NTG, FIFO, MSD, LTTG, and MTTG can be made unstable on the ring against an adversary of rate less than 1. We showed that the universal stability of FTG, LIS, and SIS queueing policies goes through in the new adversarial model, while MTTG is universally stable against a class of restricted adversaries. Furthermore, following the outline in [BKSW96] we provided a complete proof that the NTG queueing policy can be made unstable against an "adversary" that injects arbitrarily small workload at constant rate.

This work leaves open several interesting questions. One is to determine the stability of the LSD queueing policy. The LSD queueing policy gives precedence to the customer that requires the minimum service time at a given server among those that are queued at that server. Also, it would be interesting to establish whether LIFO is stable on the ring in the new model.

The stability results we presented assume that the length of the longest path of any customer is bounded. It remains an open question whether we can prove stability when the paths of the customers can become arbitrarily long (but still subject to the rate condition). Furthermore, we proved that MTTG is stable when the service requirements of any customer at any server are bounded from below, and unstable when they can become arbitrarily small.

We have seen that MTTG can become unstable in the $(0, \delta)$-model against an adversary that exploits ties among customers with the same remaining service time. We do not know if it is possible to force instability when there are no ties.

In the instability examples for the ring the same customer requires different service times at different servers, and different customers require different service times at the same server. In the proofs we fully exploit the power of the adversary in determining the service times. For the purposes of analysis of packet routing networks it would be interesting to examine whether the ring is stable in one of the following weaker adversarial models:

- Different customers may have different service requirements but the same customer has the same service requirements at every server along its path. (This models the case that different packets have different sizes).
- A customer may have different service times at different servers, but different customers have the same service times at the same server. (This models the case that different links have different service rates).
- If two different customers use the same two servers then the ratio of their service times at the different servers is the same. (This models the case that different packets have different sizes and different links have different service rates).

The above models are not trivial generalizations of the adversarial model defined in [AAF+96]. For example, breaking packets of large size into many packets of unit size, or replacing edges of large service rate with many edges of unit service rate, does not necessarily yield equivalent systems. However, it seems probable that the universal stability of the ring presented in [AAF+96] can be extended to these models. A strong indication in this direction is the proof of stability of the ring under the Cruz model, presented in [TG96], where packets of different sizes are considered.

The proof of instability of NTG against an adversary of arbitrarily small rate assumes a symmetric structure of the initial configuration. This simplifies the proof, but simulation results suggest that it is possible to relax the conditions on symmetry and still force the system to become unstable. Furthermore, simulation results indicate that it is possible to force instability when packets are generated by a Poisson process, if the system is initialized with large enough queues. It would be interesting to obtain formal proofs for these two cases. Moreover, it would be interesting to investigate if an analogous example can be

constructed for FIFO queueing policy, or if there exists some $r_0 > 0$ such that the FIFO queueing policy is stable against any adversary of rate r_0 on every network. The question was first posed by Andrews et al., for the adversarial model defined in [AAF+96], where all packets have unit service requirements at each server. In the new model, it would be interesting to examine if the example presented in [Bra94b] can be translated into the adversarial context. Furthermore, the instability example in [Bra94b] depends heavily on the immediate feedback of the servers. It would be interesting to consider the same question when we do not allow immediate feedback.

It remains an interesting open question to determine a set of general conditions that are sufficient to guarantee the stability of general multiclass networks. Recent results by Bramson [Bra96] show that the FIFO and PS queueing policies are stable under generalized assumptions on the arrival process and the service times, when the service times are class independent. It would be interesting to examine if these results can be extended to other policies, especially ones that differentiate among customers of different classes.

Finally, most instability examples consider the initial configuration of the system to be non-empty. In the adversarial model we proved that for most systems that are unstable starting from an non-empty configuration, there exists a system that is unstable starting from an empty configuration. It would be interesting to examine if it is possible to force instability on any system that starts from an empty configuration when packets are injected at constant rate.

Bibliography

[AAF+96] Matthew Andrews, Baruch Awerbach, Antonio Fernádez, Jon Kleinberg, Tom Leighton, and Zhiyong Liu. Universal stability results for greedy contention-resolution protocols. In *Foundations of Computer Science*, 1996.

[Ana96] Venkat Anantharam. Mathematical theory of communication networks. *Stochastic Analysis and Related Topics V: The Silivri workshop 1994*, pages 1–39, 1996.

[BCMP75] Forest Baskett, K. Mani Chandy, Richard R. Munz, and Fernando G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of ACM*, 22(2):248–260, April 1975.

[BF94] Francois Baccelli and Serguei Foss. Ergodicity of Jackson-type queueing networks. *Queueing Systems: Theory and Applications*, 17:5–72, 1994.

[BFM96] Francois Baccelli, Serguei Foss, and Jean Mairesse. Stationary ergodic Jackson networks: results and counter-examples. *Stochastic Networks*, pages 281–307, 1996.

[BKR+96] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. Adversarial queueing theory. In *Symposium on Computer Science*, 1996.

[BKSW96] Allan Borodin, Jon Kleinberg, Madhu Sudan, and David P. Williamson. Notes, June 1996.

[Bor86] A. A. Borovkov. Limit Theorems for Queueing Networks. *Theory of Probability and its Applications*, 31(3):413–427, 1986.

[Bra94a] Maury Bramson. Instability of FIFO queueing networks. *The Annals of Applied Probability*, 4(2):414–431, 1994.

[Bra94b] Maury Bramson. Instability of FIFO queueing networks with quick service times. *The Annals of Applied Probability*, 4(3):693–718, 1994.

[Bra96] Maury Bramson. Convergence to equilibria for fluid models of certain FIFO and processor sharing queueing networks. In *Stochastic Networks: Theory and Applications*. F. P. Kelly, S. Zachary and I. Ziedins, 1996.

[Cru91a] Rene L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[Cru91b] Rene L. Cruz. A calculus for network delay, part II: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[CTK94] Cheng-Shang Chang, Joy A. Thomas, and Shaw-Hwa Kiang. On the stability of open networks: a unified approach by stochastic dominance. *Queueing systems: Theory and Applications*, 15:239–260, 1994.

[Dai95] J. G. Dai. On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models. *The Annals of Applied Probability*, 5(1):49–77, 1995.

[DW96] J. G. Dai and G. Weiss. Stability and instability of fluid models for reentrant lines. *Mathematics on Operations Research*, 21(1), February 1996.

[Fos89] S. G. Foss. Some properties of open queueing netowrks. *Problems of Information Transmission*, 25:241–248, 1989.

[Fos91] S. G. Foss. Ergodicity of queueing networks. *Siberian Mathematical Journal*, 32(4):184–203, 1991.

[HBB94] Mor Harchol-Balter and Paul E. Black. Queueing analysis of oblivious packet-routing networks. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[HBW95] Mor Harchol-Balter and David Wolfe. Bounding delays in packet-routing networks. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, 1995.

[Jac63] J. R. Jackson. Jobshop-like queueing systems. *Management Science*, 10:131–142, 1963.

[Kel75] F. P. Kelly. Networks of queues with customers of different types. *Journal of Applied Probability*, 12:542–554, 1975.

[Kel79] F. P. Kelly. *Reversibility and Stochastic Networks*. Wiley, New York, 1979.

[KL95] Nabi Kahale and Tom Leighton. Greedy dynamic routing on arrays. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[Kle75] L. Kleinrock. *Queueing Systems*. Wiley, New York, 1975.

[KS90] P. R. Kumar and T. Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, 35:289–298, 1990.

[Kum93] P. R. Kumar. Re-entrant lines. *Queueing Systems: Theory and Applications*, 13:87–110, 1993.

[Lei90] Tom Leighton. Average case analysis of greedy routing algorithms on arrays. In *Proceedings of the Second Annual ACM Symposium on Parallel Algorithms and Architecture*, 1990.

[LK91] S. H. Lu and P. R. Kumar. Distributed scheduling based on due-dates and buffer priorities. *IEEE Transactions on Automatic Control*, 36:1406–1416, 1991.

[MD94] S. P. Meyn and D. Down. Stability of generalized Jackson networks. *Annals of Applied Probability*, 4(1):124–148, 1994.

[Mit94] Michael Mitzenmacher. Bounds on the greedy routing algorithm for array networks. In *Proceedings of the Sixth Annual ACM Symposium on Parallel Algorithms and Architecture*, 1994.

[RS92] A. N. Rybko and A. L. Stolyar. Ergodicity of stochastic processes describing the operation of open queueing networks. *Problems of Information Transmission*, 28:199–220, 1992.

[Sei94] Thomas I. Seidman. ' First Come First Served ' can be unstable! *IEEE Transactions on Automatic Control*, 39:2166–2171, 1994.

[ST91] George Stamoulis and John Tsitsiklis. The efficiency of greedy routing in hypercubes and butterflies. In *Proceedings of the Third Annual ACM Symposium on Parallel Algorithms and Architecture*, 1991.

[TG96] Leandros Tassiulas and Leonidas Georgiadis. Any work-conserving policy stabilizes the ring with spatial re-use. *IEEE Transactions on Networking*, 4(2):205–208, April 1996.