

UNIVERSITY OF CALIFORNIA
Los Angeles

Stability and Scheduling in Multiclass Queueing Networks

A thesis submitted in partial satisfaction of the
requirements for the degree
Doctorate in Philosophy in Electrical Engineering

by

Cedric Westphal

2000

The dissertation of Cédric Westphal is approved

Izhak Rubin

Thomas M. Liggett

Greg Pottie, Committee Co-Chair

Nicholas Bambos, Committee Chair

University of California, Los Angeles
2000

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 3 |
| 1.2 | Organization | 5 |
| 2 | Model development | 7 |
| 2.1 | Introduction | 7 |
| 2.2 | Review | 8 |
| 2.3 | Model | 17 |
| 3 | Rate stability | 26 |
| 3.1 | Introduction | 26 |
| 3.2 | Motivation for rate stability | 26 |
| 3.2.1 | Counterexample | 27 |
| 3.3 | Rate stability | 28 |
| 3.4 | A particular case of tandem network | 29 |
| 3.4.1 | Tandem network with forced idle periods | 31 |

| | | |
|----------|--|-----------|
| 3.5 | Proof of Theorem 2.3.1 | 34 |
| 3.6 | An example of stable idling policy | 41 |
| 3.7 | Conclusion | 41 |
| 4 | Strong stability in Markovian networks | 43 |
| 4.1 | Introduction | 43 |
| 4.2 | I.I.D Lu-Kumar network | 44 |
| 4.2.1 | Introduction | 44 |
| 4.2.2 | Initial step | 47 |
| 4.2.3 | Iterative step | 47 |
| 4.3 | I.I.D. General Network | 52 |
| 4.3.1 | Introduction | 52 |
| 4.3.2 | General Case | 53 |
| 4.3.3 | Conclusion | 57 |
| 4.4 | Harris recurrence | 57 |
| 4.4.1 | Introduction | 57 |
| 4.4.2 | Proof of the Harris recurrence | 58 |
| 4.4.3 | Fluid Model | 58 |
| 4.5 | Conclusion | 60 |
| 5 | Robustness | 62 |
| 5.1 | Introduction | 62 |
| 5.2 | Admissible parameters for a fixed random multiplexing policy | 63 |

| | | |
|----------|---|-----------|
| 5.2.1 | Introduction | 63 |
| 5.2.2 | Example | 65 |
| 5.2.3 | General case | 67 |
| 5.3 | Rate stability of the adaptive multiplexing policy | 69 |
| 5.3.1 | Markovian case | 74 |
| 5.3.2 | Conclusion | 76 |
| 6 | Application | 77 |
| 6.1 | Introduction | 77 |
| 6.2 | Background | 77 |
| 6.3 | Application of the Multiplexing Policy to a Router Architecture | 79 |
| 6.3.1 | Rate evaluation | 81 |
| 6.3.2 | Scheduling | 82 |
| 6.3.3 | Congestion report | 83 |
| 6.4 | Conclusion | 86 |
| 7 | Conclusion | 87 |
| A | Appendix | 89 |

List of Figures

- 1.1 Basic building bloc: a single server node 2

- 2.1 Lu-Kumar Network 22
- 2.2 Seidman Network 23
- 2.3 Random Multiplexing vs. Fixed Priority: Lu-Kumar network 24
- 2.4 Random Multiplexing vs. FCFS: Seidman network 25

- 3.1 Non Stationary Network 27

- 4.1 Lu-Kumar network 45
- 4.2 Extraction principle 56

- 5.1 Stability domain of a fixed random multiplexing policy 70

- 6.1 Router architecture 77

List of Tables

| | |
|--------------------------------------|----|
| 3.1 A possible sample path | 39 |
|--------------------------------------|----|

Acknowledgements

I wish to express my sincere thanks to Professors Rubin, Liggett, Pottie and my advisor, Professor Nick Bambos, for their assistance in reviewing this thesis. I would like to thank Ove Strandberg of Nokia Research Center for helping me apply my work to a practical situation.

A special word of thanks to Professor Nick Bambos, my academic advisor for all the encouragement, direction, and opportunity he has given me. His help has made possible the accomplishment of my goals.

This work was supported by the National Science Foundation Grant NCR-97 96182 and by the Nokia Research Center.

ABSTRACT OF THE THESIS

Stability and Scheduling in Multiclass Queueing Networks by

Cedric Westphal

Doctorate of Philosophy in Electrical Engineering

University of California, Los Angeles, 1999

Professor Bambos, Chair

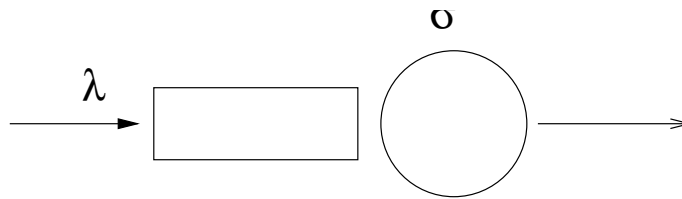
Professor Pottie, Co-chair

Chapter 1

Introduction

More and more complex organizations are modeled into queueing networks. The queueing theory, which was designed to compute the dimensions of the phone network, has extended its reach to applications into diverse fields. The basic model of one single communication link, that was captured into a single server node, has evolved into a whole array of structures interacting together. Those structures are then used to analyze organizations as varied as a communication network, customer waiting time at ATM machines at a bank, a silicon wafer manufacturing plant, a freeway system or an internet service provider domain.

The single server node model was useful for its simplicity: jobs or tasks or customers, according to the application domain would arrive according to some arrival process. Each job or task or customer would then require a service. A call in a phone system would require that a connection be established between a source and a destination through some link. A car on a freeway would require to pay for the toll. A silicon wafer would require to be baked some time in an oven. Each customer would require to spend some time using some resource: this defines the service process. When the resource is in use by another customer, new customers are buffered and wait for their turn to be served. From this setup, queueing theory would in turn be able to estimate from the arrival process and the service process some quantities of interest, as the time spent in the system, the number of customers in the system, the length of a busy period of the system or some cost measure associated with the system.



Single server node

Figure 1.1: Basic building bloc: a single server node

When customers are waiting, the node is offered a choice as to which customer to serve next. An obvious possibility is to serve customers in the order in which they arrive: this is the First Come First Served policy. Last Come First Served serves the last arrived customer in the system first. The node could also served the customer requiring the least amount of service first, or the most amount of service first, or chose randomly, or any other construction that comes to mind. By picking the customer requiring the most amount of service, we maximize for instance the number of customers in the system. We denote by scheduling the selection of the next task among those waiting by the server.

Hence, even in the very simple case of a single server node with homogeneous customers, we need to address the issue of scheduling these customers.

There are two directions in which we can extend this model: first, we can imagine nodes communicating with each other. Instead of having one node, we can interconnect several nodes in a network such that customers enter the network, visit some nodes and exit the network. For instance, silicon wafers in a manufacturing plant must be baked at some temperature at a first node, then washed at another node, then examined for defects at another node, and so on and so forth, before leaving the plant. Or a program running on some specialized computer architecture is going to require some processing time from some dedicated hardware before being processed by the CPU before being given to the graphical interface. Or a customer at a fast-food drive-in is spending some time considering what to order, then actually orders it, then waits for the burger to be ready and leaves the system. The tasks do not have to follow a single fixed sequence in the system, but can be routed in different directions according to different parameters.

Another direction in which to extend the single server model is to have a more complicated node: instead of having one server processing one type of task, we can imagine a server processing different services for different customers. The phone link for instance can be used for voice traffic or for fax traffic. The internet connection can be used for telnet traffic, or for ftp traffic. The customer at the ATM can either deposit a check or withdraw cash.

When the customers request services that vary but can be classified in different categories, we refer to multiclass traffic. A node that can handle multiclass traffic is a multiclass node. A network of such nodes is a multiclass network.

We mentioned that scheduling was an issue in a single server node, we can imagine it is of tremendous importance in as complex a situation as multiclass queueing network, where each node has to make a decision on who to serve next based on its class and its service requirement.

Queueing theory has to be extended to fit the situation of a multiclass queueing network. The complexity of the setup is such that performance measure cannot attain a high level of granularity. The level of granularity that is of interest in this work is that of stability. Without entering into any detail yet, it is intuitive to see that stability is a desired property for the examples mentioned so far. Stability is a performance measure of the network in a basic way: a good network is stable, a bad network is not. We investigate the relation between scheduling and stability.

1.1 Motivation

The issue of multiclass queueing network arose in manufacturing. If we consider a plant in which workers are able to deal with non-homogeneous products and different tools, an economically interesting question is to define a way for a worker to decide which product to work on next, so as to maximize the overall processing capacity of the plant. The historical example is that of silicon wafer manufacturing, where the same tools would be needed at different stages of production for the same wafer. The number of steps being high in the manufacturing process, and the economical pressure being strong, the decision could not be given up to intuition or primitive heuristics.

Now the set up being pretty general and abstract, there are many situations in which to apply the model and the results that can be deduced from it. An area of application is that of parallel processing in computers, in which the CPU has to split a sequence of programs into different units competing for processing at dedicated processors. With the number of processors on a chip increasing exponentially and the level of parallelisation rising as well, we need a way to schedule more and more tasks in a more and more complicated topology of processors. Of course, the fastest way is the most desirable, since the program user does not want to spend time waiting for the program completion.

Another example is that of internet routing. An Internet Service Provider manages a domain of nodes. Since massive overprovisioning of resources would be expensive, congestion is bound to happen. In order to provide the level of Quality of Service that the ISP promised to its clients, packets being routed within the domain are to be distinguished according to different classes of service. The nodes in the network have to process those packets according to their class which here correspond to some precedence level. Stability is of primary concern since packets being lost increase the traffic on the network in a cyclic pattern, or compel the client to drop its packet rate.

We restrict ourselves to these few examples, knowing that there are many other applications and it would be impossible to write up an exhaustive list. In these examples, we see that we have in common a high complexity of the structure of the network, a high diversity of the tasks at a given node, and the same need for an easy way to compute almost optimal scheduling policy.

An optimal scheduling policy would be the one that maximizes some performance measure over the set of all possible policies. The high complexity of the network structure makes such a notion of optimality prohibitive. From an engineering point of view, stability is equivalent to almost optimal: stable means that what comes in goes out, or that the throughput achieved by the network is the one set by feeding the jobs to the network.

From an engineering point of view, we also need some other properties to be added to stability: scalability, simplicity distributivity, and robustness. By scalable we mean that the complexity of computing the almost optimal scheduling policy will not get out of hand when the number of nodes in the network increases. This is particularly important in the examples we presented, since we do not want some computing time to be added to the already sensitive service time.

By simple, we mean that the policy does not require some extraneous information or some convoluted algorithms that would hinder the speed of the scheduling decision.

By distributive, we mean that a "good" scheduling policy would only require local information, and be independent of the past and future of the customers in the network to achieve sub-optimality. This is linked to the previous two notions, since a local algorithm is more easily scalable, and is more simple than an algorithm taking into account many more variables.

By robust, we mean that a small error in the parameters of the system would not impede the functioning of the network. Even better would be a self-adapting algorithm that would filter out the possible errors in the parameters.

Those characteristics are motivated by the last two examples. In a silicon wafer manufacturing plant, the time frame is such that it is possible to analyse carefully off-line the next decision to be taken, and to refine an algorithm off-line. In the last two examples, the processing times are of the scale of the decision time. Hence, a long decision designed to ensure stability would lead to the exact opposite behavior. Also for the example of the ISP, total knowledge of the network can be impossible, since part of the network can be in another domain. A centralized intelligence making the decision would need some signalization, which is of the same nature as the traffic in the network. Hence in this particular case, a centralized decision would add traffic in the network and as a consequence increase the risk of congestion.

This illustrates the varied applications of the model as well as the need for a stable, distributed, scalable and robust scheduling policy.

1.2 Organization

In the following, we will first define the model in a more detailed manner. We will give more background on multiclass queueing networks and explain why obvious scheduling policies such as fixed priority and First-Come First-Served do not satisfy the desired requirement. This will be the object of the chapter 2

In Chapter 3, we will answer the following question: what is the stability region of any multiclass queueing network? It is obvious that each station in the network can process at most one unit of work per unit of time. Now interaction between nodes and interaction between the classes could shrink the stability domain. By starving some nodes while processing some workload in another part of the network, or by creating blocking modes or resonant oscillating patterns, the network interactions should intuitively restrict the possible arrival rates of customers. We will prove that it is not so, and that for any network topology, it is possible to construct a scheduling policy such that not only the stability domain is optimal, but also the scheduling policy we introduce satisfies the desired requirement of scalability, distributivity and robustness.

In the chapter 4, we will extend the stability result to a stricter definition of stability. Rate stability does not ensure that the number of customers stay finite, it ensures that what comes in goes out in a timely fashion. We will prove that in a Markovian network, where interarrival times and service times are random variables with general yet independent and identical distributions, the number of customers is bounded by a random variable that is

finite almost surely, and so is the time spent in the network by any customer. We can also explicitly compute the bound and compare it to a quantity extracted from a single server node. This result is extremely important, since it says that even though there is interaction between classes and between nodes, we can separate a node from the surrounding network using the scheduling policy we defined.

In the chapter 5, we address the case in which rates might be varying in time. We investigate the stability of a fixed random multiplexing policy with respect to small variations of the parameters and compute the stability domain of a fixed scheduling policy. We will present as well a policy that adapts itself to the network environment. Basically, this means that a node being able to only differentiate the classes is able to construct from scratch a scheduling policy that is stable. Hence by plugging such a black box into a network, that knows only how to process the different classes, but has no other information, the network would still be stable.

Eventually, in the last chapter 6, we will consider the reverse problem in the framework of an application. The reverse problem is the following: knowing that a given class must get a fixed ratio of service with respect to another class whenever these two classes are competing for resources (for instance, two customers of class 1 must be served for every one customer of class 2), what is the stability region, or what incoming rates for these classes can the system accommodate. It is the reversed problem from 3 since in that chapter, the stability region was given (to be the maximal possible) and the definition of the scheduling policy implied to compute some ratio for precedence (so as to ensure flow conservation).

This last problem is related to the differentiation of service over the internet. In the framework of their service agreement, the ISP will guarantee its client that packets marked as priority one will get say twice as much service as priority two. Now to enforce this guarantee, the ISP needs to know how many of packets from each class it can accept in the network, hence the need to define some stability region.

We will offer some concluding remarks in Chapter 7, as well as some directions for future work.

Chapter 2

Model development

2.1 Introduction

In this chapter, we present and motivate our underlying model. We give a brief historical perspective in order to better understand how our model arose and why it is relevant. Also, by taking a couple steps backward in time, we can better build some intuition on the subject. This will be the object of the next section. We will then introduce the model that will we study in the rest of this document. This model description corresponds to the trunk of our organizational tree, the branches of which are the next chapters and results. This is the reason why it requires its own chapter. In the last section, we will briefly state these results using the framework defined here.

A quick point of vocabulary: we use indifferently the words node, server, station to denote a node in the network. We use also the words customer, task, job, call to describe the same objects. Vertices in the graph of the networks are denoted channels, links, routes. As it could be expected, the variety of the applications gave rise to the same variety in the denominations.

2.2 Review

As we mentioned already, the field of applications of our work is lying in between computer systems, telecommunication networks and manufacturing. As a consequence, most of the works in the literature on this topic are pretty recent. We cannot go back in time any further than the Industrial Revolution after all. To our knowledge, the first to investigate the topic of communication networks was Erlang [28], [29] who computed the probability that a phone user will find a line busy when trying to place a call. This was the first mathematical model of a phone network, in which several users share the same resource. It had some intrinsic properties that made the model tractable: phone calls arrival processes are well approximated by Poisson processes, and exponential distributions fit the call length distributions. In today's terms and in Kendall notation, and oversimplifying, Erlang was studying a M/M/1 queue.

We will skip a few steps along the way, and mention two milestones which pave the way to the present research area:

Jackson [38], [39], while considering some manufacturing applications, took the M/M/1 node, and placed it into a network. In his model, jobs would enter node s in a system of servers $\{1, \dots, S\}$ according to a Poisson process with some rate depending on the node s . The customers entering node s would be placed in the input buffer of node s and wait for service in a First-Come First-Served basis. Then, the jobs would receive an exponential service time with distribution depending only on the node s . After service completion, it would be sent to a node u with probability $p_{s,u}$, be buffered again at node s with probability $p_{s,s}$ or leave the network with probability $1 - \sum_{u=1}^S p_{s,u}$. All service times and all interarrival times to a node are independent. Note that from the node point of view, all the customers are undistinguishable, since their service times are independent of past events and set by the node itself.

In this situation, Jackson was able to identify the convergence of the state of the system to a stationary distribution, independently of the initial conditions. Moreover, the stationary distribution is the product of the distributions of the state of a single node extracted from the network, when fed with the proper incoming Poisson process. It follows that the issue of stability is easily resolved in a Jackson network: if the actual load at each station, traditionally denoted by ρ , is strictly less than one, then the network is stable. If not, the number of customers in the system grows almost surely to infinity. The actual load is computed by inverting a matrix of dimension equal to the number of nodes in the network.

One proof is to write down the transition matrix for the Markov chain associated with the network, and to check that the product form solution is indeed an invariant distribution. Uniqueness of the invariant distribution is assured by classical results on Markov processes.

Stability domain: As we all know, the condition $\rho < 1$ at each node is a necessary stability condition. ρ , namely the ratio of the arrival rate by the service rate, is a function of the arrival rate λ and the service rate μ . Thus the conditions $\rho < 1$, $\mu \geq 0$ and $\lambda \geq 0$ define a volume in the space of all possible λ 's and μ 's.

The stability domain of a network is the set of values that can take the arrival rates to a network and the service rates within the network, so that the network is stable. Thus, the stability domain of a network has to be included in the area delimited by $\rho < 1$. Namely, in the space of all possible λ 's and μ 's, the stability domain has to be included in the volume $\rho < 1$.

We can rephrase Jackson's result as: a Jackson network maximizes its stability domain. The volume $\rho < 1$ and the stability domain coincide.

Actually, a proof that $\rho < 1$ at each node is a necessary stability condition is given in our other milestone. Loynes [53] considered a single node with FCFS service, but with arrival processes and service processes that were stationary and ergodic instead of being independent and identically distributed with exponential distribution. By an original construction, he was able to prove the convergence of the departure process of such a node to a stationary and ergodic one, which was stable (in his term *honest*, that is finite almost surely) for $\rho < 1$.

We want to detail a little bit more Loynes' construction, since we will use a similar construction to prove our stability result in chapter 4. The idea is to consider the waiting time of customer 0 when we let in customers $-N, -N + 1, \dots, -1$ and 0. Basically, this waiting time increases with N , and thus the sequence of waiting time defined this way converges as N goes to infinity. Thanks to the stationarity and ergodicity of the driving sequences (the interarrival process and the service times), Loynes was able to retrieve the same properties in the limiting process of the waiting times as N goes to infinity.

The same construction could be extended to the case of an acyclic network. As long as there is no feedback in the network, the monotonicity of the waiting time sequence with respect to the number of customers we let in is preserved.

Stability. We have used the work *stable*, using its intuitive meaning. As illustrated in the two previous examples, stability is an asymptotic property. We let the system run for a very long time, and if there are few customers in the system, then it is stable. We cannot expect

the number of jobs in the system to be bounded: a single M/M/1 queue cannot offer such a guarantee. We could require on the other hand the number of customers divided by time to go to zero. This is our definition of stability:

Definition 2.2.1 *A system is stable if the number of customers at a time t divided by t goes to 0 almost surely as t goes to infinity.*

Another candidate for stability is that the number of customers is finite almost surely as time goes to infinity. Of course, this is included in the previous definition, and we refer to this stability as *strong* stability.

Scheduling. Scheduling is the way to order the task for service at a node. In a Jackson network, the service depends only on the station, so there is no scheduling issue there. From the server or the router point of view, all customers are identical. There is no issue of scheduling as far as stability is concerned in a single server queue either, since no matter how we shuffle the workload, it is the same in a busy period. Scheduling becomes primordial however when dealing with multiclass networks.

Generalizing the reach of Jackson's model was the goal of many references. Baskett et al. [8] dealt with multiclass Jackson networks and different scheduling policies. Multiclass networks are networks for which the arrival rates and the service requirements depends on the class of the customer. Multiclass means that at a same node, customers of different types will receive service according to different statistics. The queueing policies they investigated were processor sharing, preemptive/resume Last Come First Serve, and FCFS with iid exponential service times. Kelly [40],[41] also considered network of multiclass customers with exponential service times. Baskett et al. and Kelly obtain product form solutions for the steady-state distributions: namely, each node can be extracted from the network and studied by itself. The main property they identify is that of reversibility [42]. Reversibility means that the output process considered when reversing time is of the same nature as the input process. The necessary condition $\rho < 1$ is also obviously sufficient whenever there is a product form solution.

A single class generalization of Jackson networks to ergodic and stationary input process can be found in [1]. Once again, stability is proven under the condition $\rho < 1$. Other studies of open queueing networks under general distributions and with single class could be found in [11], [61], [57]. Recently, the focus has turned onto multiclass networks.

The more complicated the network, the more stability starts being a performance measure of interest for the network. A stable network is one in which the output rate is maximal,

since it is equal to the input rate. From an engineering point of view, it is often sufficient to know the stability of the network to design the network.

Multiclass. In a multiclass queueing network, the scheduling policy is of tremendous importance. The throughput of such a network is highly dependent on the queueing policy. Since each server receives jobs belonging to different types, each server has to make a decision whenever facing a choice as to which class should be served next. Now the overall throughput of the network depends on whether this decision is bad or good. It can happen that a flawed decision policy starves some stations when other process workload. This creates *blocking modes* in the network, that is patterns which prevent the stations to work harmoniously and together in parallel.

An example of such patterns is given in [4] for a closed 2-class cyclic network with First Come First Served policy. Bambos shows that different initial conditions lead to different throughput. The blocking mode is sustained because the two customers of different statistics (we could say of different classes) cannot overtake each other. The choice of FCFS in this setting is detrimental to the optimal performance of the network.

Some oscillating patterns could also occur due to the structure of the network and the statistical requirements of the customers. If the network enters a resonant mode, it leads to instability, even though the networks empties every so often [52].

Several scheduling policies come naturally to mind. It depends only on the imagination of the reader to create as many as possible. Some features must be satisfied for a queueing policy to be admissible though.

Work conserving. In order to minimize the workload in the system, we consider only *work conserving* or *non-idling* policies. A work conserving policy is a policy in which a station never idles whenever there is workload waiting to be processed at this station. It cannot for instance wait for some customer of a different class to arrive, when there are already customers awaiting service. We will actually introduce in the sequel an idling queueing policy which is stable. This requirement is due to common sense more than mathematical necessity. It is beyond our awareness that an idling scheduling policy would outperform all non-idling policy. Let it be a philosophical axiom that a server should start working right away and never ever procrastinate.

We will only consider scheduling policies which are class based, and not customer based. That is, for each customer of the same class arriving at a given station, their processing order is First Come First Served. This requirement is a necessity when the network grows and the number of customers as well. A customer based queueing policy would keep track of

an increasing number of parameters and not be scalable. The complexity of the optimization would get out of hand.

The scheduling policies we consider are also predictable. By predictable we mean that a station cannot base its scheduling decision on future events, but only on the information $\mathcal{F}_t, t \leq t_o$ at time t_o , where \mathcal{F}_t is some filtration corresponding to the network state. This has to be somehow related to the fact that we consider non-idling policy. It could make more sense to wait for some customer to enter the system if we knew its service requirement. Note that since we have not assumed the independence of customers, there is some information available on the future in the filtration up to time t_o .

In our phrasing, we have been implying that the station was making the decision. It is a bit abusive, since we could imagine scheduling policies being network-wide. Since a customer will travel through the network, it is not obvious that a blind decision at the server level would be a good decision. Actually, most policies of interest are distributed, namely the station decides on the class to be served next. It is a feature of interest for a scheduling policy to be distributed, since it reduces the computation level to make the decision, and is easily scalable in case of modification of the network.

Bambos illustrated this scalability issue in his investigation of the relation between stability and scheduling in a class of processing systems [6]. In a parallel processing system with precedence constraints, he could identify the optimal scheduling policy, which necessitates the resolution of a linear program for a batch of customers. In this case he was able to prove also that the complexity of solving this program did not grow linearly with the number of customers, hence that the optimal scheduling was feasible. Yet the processing system could not be extended to the case of networks with feedback.

Examples. For given input streams into the network, we would like to investigate the issue of stability. It is obvious that we need the average load per unit of time at each station to be less than one. This condition is not sufficient in many cases. A classic example of instability is the Lu-Kumar network (cited in [47]). It is a 2 station 2 classes network illustrated in fig. 2.1.

Customers enter the network at rate 1, and are served respectively a time $\sigma_1 = 0, \sigma_2 = 2/3, \sigma_3 = 0, \sigma_4 = 2/3$ at buffer 1,2,3,4. The service times are deterministic, as well as the arrival times. Station I could decide to give higher priority to customers of class 4. Since $\sigma_4 > \sigma_1$, the station would empty the workload at the maximal rate from its buffer. If station II, for the same reason gives higher priority to class 2, the network is unstable (cf. fig.2.3). By unstable, we mean that the number of customers in the system at a given time increases to infinity as fast as time.

This could seem to be a contradiction with our wish that a scheduling policy be distributed. Station I and station II do blindly what is best for them, get rid of as much workload as they can as fast as they can, and end up with more than they can deal with.

To illustrate what is going on, consider we start at time 0 with N customers in buffer 1 of station I. Since $\sigma_1 = 0$, at time $0+$ these N customers are immediately placed in buffer 2. Since buffer 2 has a higher priority than buffer 3, these N customers and all the subsequent arrival are served before any of the customers in buffer 3. The arrival in the system find buffer 1 empty, then zip through station I, and are placed at the end of buffer 2. It takes $2N/3$ units of time to serve the N initial customers, during which time $2N/3$ new customers arrive, and so on and so forth, until time $2N$, at which time station II starts serving customers in buffer 3. At time $2N$, we have of course N initial plus $2N$ new customers in the system. These $3N$ customers take no time to get served at station 2 since $\sigma_3 = 0$, hence, at time $2N+$, we have $3N$ customers waiting for service at buffer 4, station I.

Since their service time is $2/3$, it takes $2N$ units of time to process them all. During this time, since buffer 4 has a higher priority than buffer 1, all the arrival are stuck at buffer 1. Thus, at time $4N$, we have $2N$ customers in buffer 1 and a network which is otherwise empty. That is, the number of customers has doubled in the system. Now, since we are in the same configuration as initially, this pattern will repeat until the load on the system becomes unmanageable.

There exist in this example a sequence of times, say T_n , such that the number of customers in the system at time t , say $N(T)$ Satisfies:

$$\frac{N(T_n)}{T_n} \rightarrow \infty \quad (2.1)$$

This contradicts our notion of stability.

The most natural policy, the First Come First Served policy has also been proved to be unstable in this framework [60], [12], [13]. In figure 2.2 is pictured the Seidman networks, where the load in the network under certain initial condition increases linearly with time. Bramson is even able to construct a network for which the load at each node can be taken as small as desired, and yet, using FCFS as a scheduling policy leads to instability.

In a deterministic setting, Kumar [51],[48],[52] et al. have studied networks with a single fixed route for the customers. In such a re-entrant line, as these networks are called, First-Buffer First Served and Last Buffer First Served are stable. Those are fixed priority

disciplines, that assign the higher (respectively the lower) priority to first buffer a customer is visiting along its route.

Different techniques have been devised to evaluate the stability of a given policy. In the case of a Markovian network, namely when the interarrival times and the service requirements of customers from a given class are independent and identically distributed, a fluid limit may be attained for the number of customers in the system (see [19], [20], [16], [9]). In this Markovian framework, it has been proven that feed-forward, cyclic, 2-stations with one station tending only one class are stable under any work conserving scheduling policy. It has also been proven that in all re-entrant lines (ie. a network where all the customers follow the same given route [47]) Last Buffer First Serve was stable. The stability discussed in the Markovian framework is that of Harris recurrence.

Harris recurrence: Harris recurrence of a process $X_t, t \geq 0$ satisfying the strong Markov property means that, for a given set A and an initial condition x , if there exists a σ -finite measure μ on the state space of X_t such that whenever $\mu(A) > 0$, then the stopping time $\tau_A = \inf\{t \geq 0 : X_t \in A\}$ is finite almost surely. Namely, we can reach any state with non-zero μ -probability in a finite time. A consequence of this (see [30]) is that there exists a unique invariant measure; if this measure is finite, then the network is strongly stable. The time spent in state A is also infinite with probability 1. This result depends heavily on the Markovian property of the process X_t . Independence of service times is usually a dream, and the complexity of the description of the system (that needs to include a lot if not all the information about the state of the system for it to be Markov) prevents the use of this method in many network topologies.

Also in the Markovian framework, we find in [54], [55] a scheduling policy which is stable independently of the network topology. Yet, this policy is not distributed, and requires the solution of linear programs growing in size with the number of buffers in the network. It is not applicable in a communication network for instance, where no central intelligence exists that could make the scheduling computations. Furthermore, the signaling to the central scheduler would add resource use to many applications, that itself could render the network unstable.

Another field of study that has grown tremendously recently is that of diffusion approximation for multiclass networks. Markovian fluid models, or rate-stability results are first-order results. The purpose is to scale some process X_{nt}/nt and let n go to infinity. If for say $t = 1$, the limit process $\tilde{X}_1 = 0$ then the network is stable. Any condition on \tilde{X}_t would be a first-order condition, involving the mean of the input processes. Diffusion approximation uses a different scaling of time and space, X_{nt}/\sqrt{nt} , so as to conserve the second moment

information. We refer the reader to [32]-[35],[63]. There is still some questions about which scaling to use, and which topologies are suitable for such a treatment, and practical results of interest have been restricted to very simple networks.

In the stationary and ergodic framework, Bambos and Wasserman [5], [3] studied some variations of a multiclass pipelined network, with or without feedback. They considered a static case, in which customers were present at the first node of the network at time 0, and introduced a policy that optimized the time to empty the network. The policy was as follow: assume that at time 0, N_k customers of class k are present. There are K different classes ($1 \leq k \leq K$). Whenever the first node of the network completes the service of a customer, it picks the next class from which to pick a customer by tossing a coin pointing at class k with probability $N_k/(\sum N_i)$. Wasserman and Bambos were able to prove that this way of choosing a customer is asymptotically optimal, when the N_k 's increase to infinity while keeping their proportions constant.

While the result itself is not important (Loynes proved that any ordering of these customers would yield the same throughput in a tandem network; the point of the paper was in its geometrical proof), it is this idea of mixing the customers in a random way, so as to avoid the building of blocking patterns, which triggered the present work.

Definitions of Stability Domains: A question of interest in a couple of references [22], [24] is the notion of stability domain. A network is a set of parameters on a given space. If we consider the averaged arrival rates and service rates, we can map a network to some multidimensional space. Assume for instance that we normalize the arrival rates, and that we have N different types service requests within the whole network: we can map our network on a N -dimensional space $(\bar{\sigma}_1, \dots, \bar{\sigma}_N)$.

Under the assumption that stability depends only on the first moment of the service random variables and of the interarrival rates, we can then define a stability domain for a network.

Consider for instance a single server in which two flows with arrival rate normalized to 1 request service with mean service time $\bar{\sigma}_1$ and $\bar{\sigma}_2$ respectively. The total load arriving the system on average per unit of time is $\bar{\sigma}_1 + \bar{\sigma}_2$, which has to be less than 1, This last condition defines a simplex on the 2-dimensional space $(\bar{\sigma}_1, \bar{\sigma}_2)$. In this very simple situation, it is obvious that for any scheduling policy ordering the two classes 1 and 2, the stability domain is the same, and coincides with this simplex.

We have seen with the Lu-Kumar network that in the 4-dimensional space $(\bar{\sigma}_1, \bar{\sigma}_2, \bar{\sigma}_3, \bar{\sigma}_4)$, that the set defined by the necessary conditions:

$$\bar{\sigma}_1 + \bar{\sigma}_4 < 1 \text{ and } \bar{\sigma}_2 + \bar{\sigma}_3 < 1 \quad (2.2)$$

does not coincide with the stability domain for some Fixed Priority scheduling policy.

A question of interest is that of the global stability domain: namely, for which parameters $(\bar{\sigma}_1, \dots, \bar{\sigma}_N)$ all the scheduling policies are stable. Dai et al. ([22],[24]) discuss this in the case of a 2-stations and 3-stations networks. For instance, in the Lu-Kumar network, if we add the equation:

$$\bar{\sigma}_2 + \bar{\sigma}_4 < 1 \quad (2.3)$$

to the equations 2.2, Dai proves that for any scheduling policy this network is stable. This is denoted as the global stability domain: it is the domain in which no matter how hard you try to drive the network away from the stable state, you cannot. It is particularly interesting to define in some complicated network environments where it is critical that the network behaves in a stable manner.

We mention this to differentiate our work from these results: we mean by stability domain in the sequel of this work that there exists at least one stable scheduling policy. If we denote by \mathcal{D}_ϕ the stability domain corresponding to a scheduling policy ϕ , by $\mathcal{D}_{\text{global}}$ the stability domain for all policies, by RMP the scheduling policy that we will define later on, and $\rho < 1$ the necessary network-wide stability condition, then:

$$\mathcal{D}_{\text{global}} \subset \mathcal{D}_\phi \subset \mathcal{D}_{RMP} = \mathcal{D}_{\rho < 1} \quad (2.4)$$

where we will prove the later equality in the sequel. It is interesting to note that an intuitive property such as: $\mathcal{D}_{\text{global}}$ *is convex* is not true. It is possible, and an example is given in [24], that for some service times, the network is stable, and by reducing some of the service times, we drive the network to unstability.

This illustrates the importance of a scheduling policy that would yield this stability and maximize its stability domain.

As we mentioned earlier, stability is an asymptotic event. A question of interest would be to study the transient phase that lead to the stable phase. Namely, study the dependency of the network state on the initial condition. This will not be addressed in this work. It is a topic of further research.

Organization of the remainder: In the next section, we will discuss our model of multiclass queueing networks, and define a policy which ensures the rate-stability of any topology of network. This policy is distributed, predictable, computable, and requires minimum knowledge about the input stream to a given station. To illustrate this claim, we depict in figure 2.3 and 2.4 the load in one buffer of the Lu-Kumar network, and the Seidman network respectively.

2.3 Model

The network setup is different from the classical network depicted in [19] for instance. The reason for the difference is that we want the customer entering the system to *carry all the information*. In a network where the service times accorded to a class of customers are i.i.d., customers from the same class are undistinguishable. In our setup, customers bring in the amount of work they require as well as the route through the system they want to follow.

We consider a network composed of B buffers. We denote by $\mathcal{B} = \{1, \dots, B\}$ the set of the buffers. We consider two partitions of the set \mathcal{B} , the partition s_1, s_2, \dots, s_S and the partition r_1, r_2, \dots, r_R . S is the number of stations in the networks, s_1, s_2, \dots, s_S are the S stations in the network. R is the number of routes in the network, and r_1, r_2, \dots, r_R are the R possible routes in the network. Of course, by definition, $S \leq B$, $R \leq B$.

Each buffer feeds customers to a given server of a given station. Namely, buffer $b \in \mathcal{B} = \{1, \dots, B\}$ gives customers to station $s_b \in \mathcal{S} = \{1, \dots, S\}$. s_b is the station corresponding to buffer b . The relation $b \rightarrow s_b$ is a projection of \mathcal{B} onto $\{s_1, \dots, s_S\}$. Each buffer b correspond to a class at station s_b . Since several buffers may share the same station, this is a multiclass network. This implies that a station will have to make a decision on which customer to serve whenever several of its buffers are non-empty. Our purpose is to propose a decision making policy which stabilizes the network.

A route in the network is a sequence of buffers visited in increasing order. It indicates which buffers (and consequently which stations) a customer following this route must visit, and in which order. A route cannot visit twice the same buffer, and no two routes can share the same buffer. Also, every buffer must be part of a given route. Every buffer belongs to one and only one route. We denote by \mathcal{R} the set of possible routes.

We can view the partition \mathcal{S} as a vertical partition of the set of buffers, and \mathcal{R} as an horizontal one. A route could visit the same station several times, hence the notion of class

cannot be associated with a route, or with a customer. A customer change classes whenever it moves from a buffer to the next one on its route.

The route r visits v buffers, say (b_1, \dots, b_v) . The j th customer in class $r \in \mathcal{R}$ is a sequence $(t_j^r, \sigma_j^{b_1}, \dots, \sigma_j^{b_v})$, defining the arrival time and the service time requirement at each step along the route. $\tau_j = t_j - t_{j-1}$ is the interarrival time. The total input to the system by the j th customer of each route is thus a n -tuple

$$IN_j = \left(t_j^{r_1}, t_j^{r_2}, \dots, t_j^{r_R}, \sigma_j^{b_1}, \sigma_j^{b_2}, \dots, \sigma_j^{b_B} \right) \quad (2.5)$$

We assume that the sequence IN_j is stationary and ergodic with respect to the j -index shift.

Customers arriving in the system are buffered in the first buffer of their route, until they get admitted into service. Upon completion of their service requirement, they are buffered in the next buffer on their route. When there is no next buffer according to their routing table, they leave the system. Each *buffer* follows a first in, first out policy. As we required earlier, the scheduling policy has to be class (or equivalently buffer)-based.

We define some quantities that we will need all along the sequel of this document. We denote

$$\tau^r = E[\tau_1^r]; \lambda^r = \frac{1}{\tau^r} \text{ and } \bar{\sigma}^b = E[\sigma_1^b] \quad (2.6)$$

for all buffers $b \in \mathcal{B}$. λ^r is the traffic intensity of route r and $\bar{\sigma}^b$ is the mean service requirement at buffer b . Since a buffer $b \in \mathcal{B}$ belongs to one and only one route $r \in \mathcal{R}$, we can define $\lambda_b = \lambda^r$.

Remark: this setup does include random routing, if the number of visits to a given station is bounded, and the routing is independent of the service and arrival times. Indeed, if the number of visits to a station is bounded by a quantity V , since we have S stations, there are at most SV steps for a customer visiting the network. Each one of these SV steps is chosen from a subset of the S stations (namely the stations that have not yet been visited V times by the customers), hence there are less than $(SV)^S$ possible routes.

While this number can be very large, it is still finite, so we can enumerate the routes, and distinguishing theoretically among the customers. Due to the independence of the routing from the other random sequences, we can assume that the routing of a customer is known ahead of time, and thus assign the route to the customer as part of its information upon arrival. We can then compute the flow of the route, and all the subsequent quantities we need to derive from it. Recall that a customer carries its service information, and is not assigned a service time by the station.

Another remark: It also includes the dependence of the routing of a task upon the service time requirement of this task. Also a case of practical application is the case when the set of routes \mathcal{R} is a singleton, in which case our network is a re-entrant line already mentioned [47].

This setup could be extended with no effort to the case of an infinite yet countable number of routes such that these routes require only a finite number of different service time statistics at a given station. We could not define the set of routes as a partition of the set of buffers anymore: we would need to aggregate all routes requiring the same statistics at a given station in the same buffer. Also, customers would be given their service time by the station. For instance, this setup includes Jackson networks.

In the sequel, whenever we mention a network, it is assumed to satisfy this model.

A necessary condition for the stability of the network is that the load brought to each station per unit of time is less than one. Namely, we need:

$$\forall s \in \mathcal{S}, \quad \sum_{\{b:s_b=s\}} \lambda_b \bar{\sigma}^b < 1 \quad (2.7)$$

We will not prove this necessary condition, it is a classical and obvious result. Usually, we define ρ at a given node to be (without writing the dependency of ρ on the station s):

$$\rho = \sum_{\{b:s_b=s\}} \lambda_b \bar{\sigma}^b \quad (2.8)$$

This condition is of course not sufficient, as we have already discussed. The stability of the network also depends on the choice made at each station to serve a customer from one of its buffers over another customer from another buffer. We denote by Φ the set of all possible decision-making policy which are non idling or work conserving. This is a set of very natural policies, since they do not add any load to the system but the one requested by the customers. Some idling policies could keep the system stable, but would hinder the performance of the system in a lightly loaded situation. We know that some policies such as First Come First Serve, or Last Come First Serve, or Fixed Priority policies, which all belong to Φ , do not ensure the stability of the network for all possible network topologies under condition (2.7) (cf. [47], [51], [20],[12], [13], [59], [60], [57]).

We now define a policy which we call the Statistical Multiplexing policy. This policy is a dynamic equivalent of the policy mentioned while discussing[3]: dynamic since it uses ratios that are computed not from the quantities of customers in the system at time 0, but from the flow of this customers in the station, were that station stable.

It is a distributed policy, meaning that each station proceeds independently of the other ones. Consider station s . Denote $B_s = \{b \in \mathcal{B} : s_b = s\}$, the set of buffers connected to station s . The total flow of customer in server s per unit of time if the system is stable is $\lambda^s = \sum_{b \in B_s} \lambda_b$. We define, for all $b \in B_s$:

$$\pi_b^s = \frac{\lambda_b}{\lambda^s}. \quad (2.9)$$

Now, at each decision epoch, station s throws a coin pointing at buffer $b \in B_s$ with probability π_b^s . If the buffer b pointed at is non-empty, the first customer arrived in this buffer goes into service. The next decision epoch is then the completion of this customer's service. If the buffer b pointed at is empty, the station tosses the coin again, and again, until it points to a non-empty buffer. If all the buffers are empty, then the next customer arriving to station s goes into service. We assume that the action of tossing the buffer-designating coin takes no time. It is a reasonable assumption whenever it is neglectible with respect to the service times of the customers. As we can see, this policy is work-conserving and distributed. It is simple, since the computation time needed to decide which customer is next is not function of the size of the network or the number of customers in the network. We will prove that it ensures the stability in the system whenever the stability equation 2.7 is satisfied.

Theorem 2.3.1 *The stability domain of any network is the domain $\rho < 1$ defined in 2.7*

More precisely, we will prove in chapter 3:

Theorem 2.3.2 *The random multiplexing scheduling policy is rate-stable for any network topology under the necessary stability condition.*

In chapter4, we will restrict the hypothesis to the case of i.i.d. sequences of general distribution, and we will extend this result to prove that

Theorem 2.3.3 *The strong-stability domain of any Markovian network is the domain $\rho < 1$; or, the random multiplexing scheduling policy is strongly stable for any network topology.*

where strong stability will mean either an almost surely finite distributional bound on the waiting time of any customer, then an almost convergence to steady state of the Markov process describing the system.

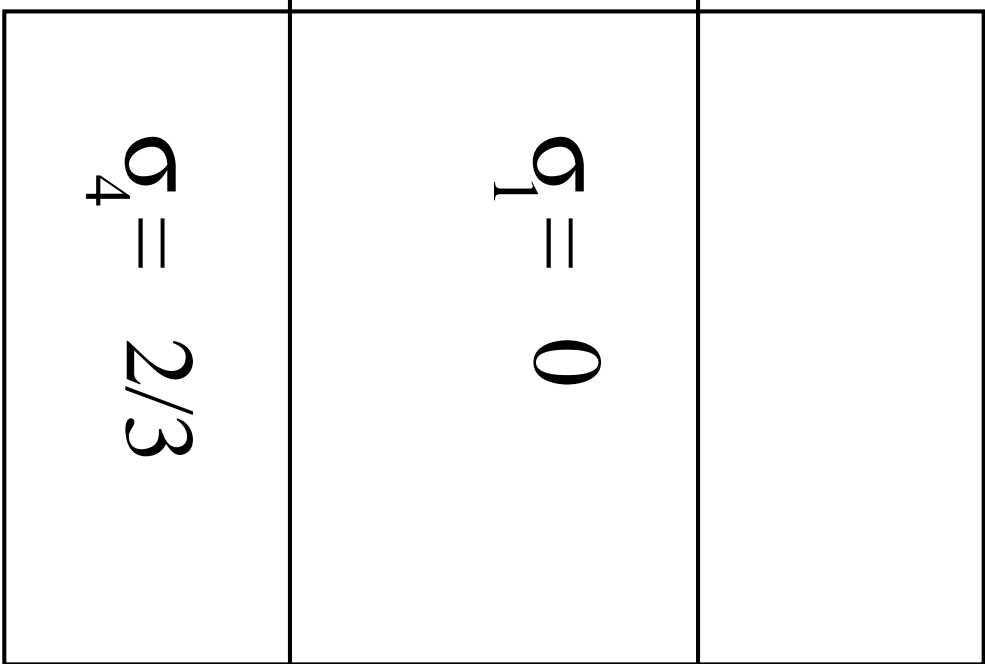
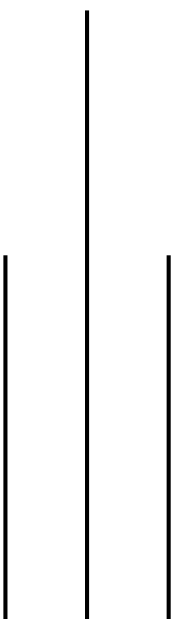
In chapter 5, we address the issue of the robustness of such a policy, in two ways: we prove that small perturbations in the service rates or in the arrival rates do not jeopardize the stability in the system. The random multiplexing policy is also stable in the sense that it is insensitive to small variations of the parameters. Also, in the case of bigger perturbation, we introduce a scheduling policy which is adaptive. Namely, assuming the routing is fixed, the scheduling policy we introduce would overcome any change in the arrival flows.

Theorem 2.3.4 *The adaptive random multiplexing scheduling policy stabilizes any network topology under the necessary stability condition*

In chapter 6, we consider the reverse problem. Instead of proving that for any point in the domain $\rho < 1$ we can construct a stable queueing policy, we look at the stability domain of one singular scheduling policy. Namely, if we give ourselves a scheduling policy taken from the class of the random multiplexing policies, for which values of the parameters will this policy be stable for the network. The application of interest is that of a router in an IP network.

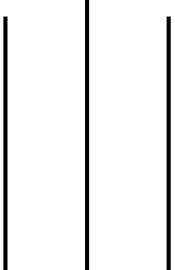
This last question is of tremendous importance, since we might have a contract agreement with a client such that its customers are to receive twice as much service as those of another client. For instance, a client A would agree to pay a premium to have its tasks taken care of more quickly than the tasks of a client B not paying the premium. The stability domain in this situation tells us at what rate we can accept request for service from client A and client B. The model will stay the same, independently of the point of view we consider.

buffer 1

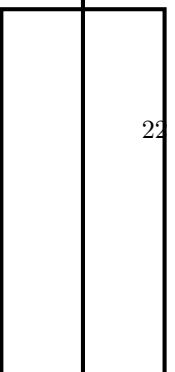


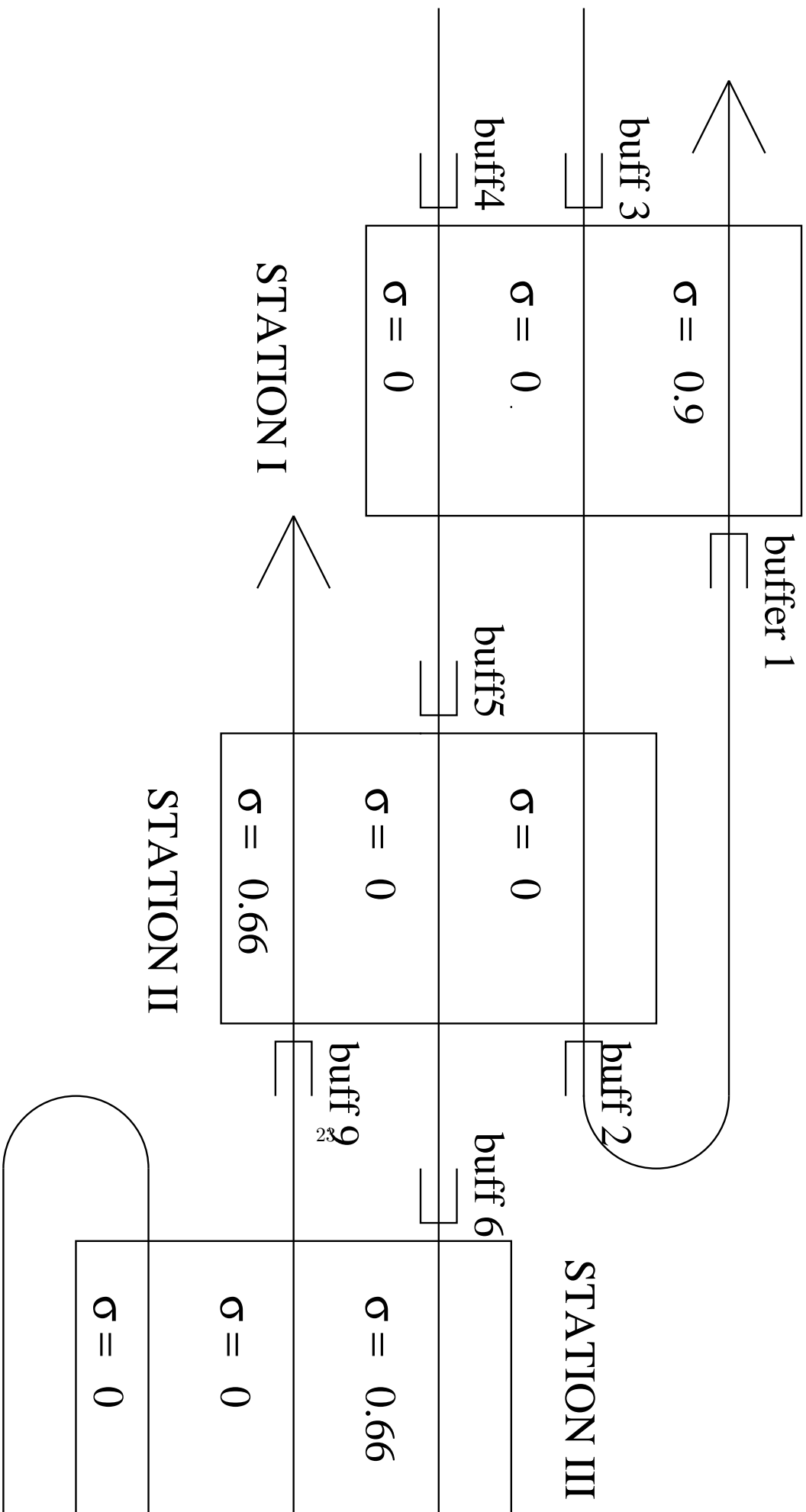
STATION I

buffer 2



buffer 4





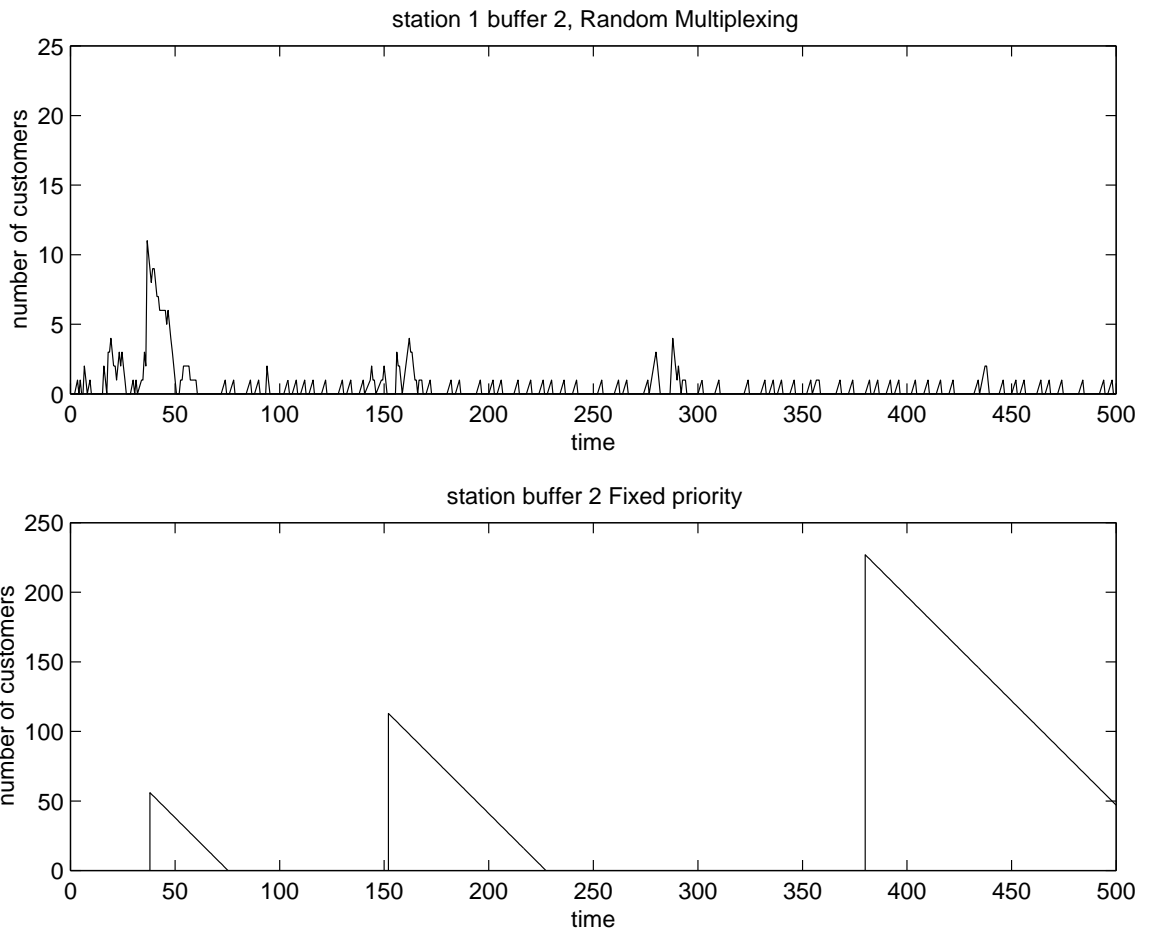


Figure 2.3: Random Multiplexing vs. Fixed Priority: Lu-Kumar network

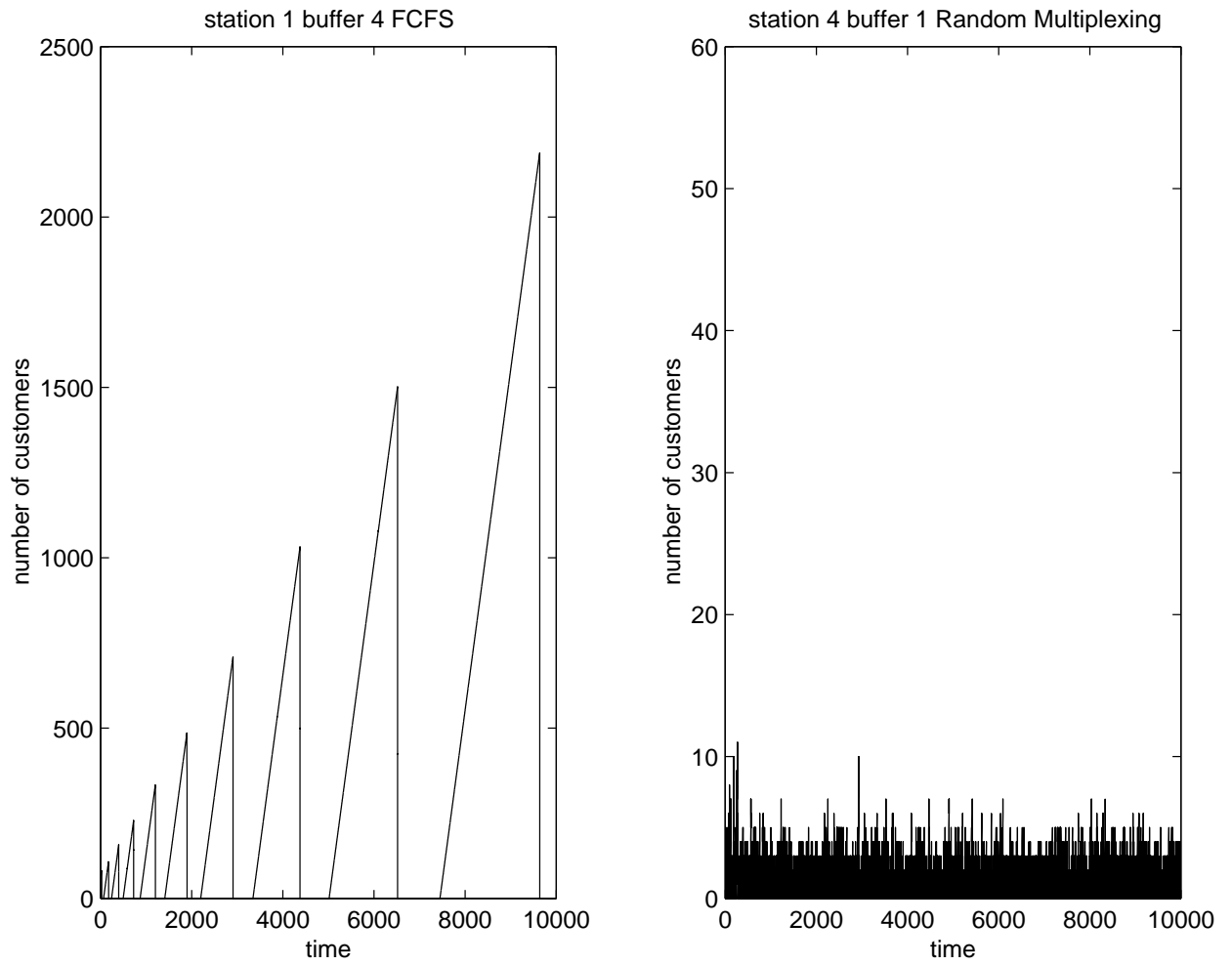


Figure 2.4: Random Multiplexing vs. FCFS: Seidman network

Chapter 3

Rate stability

3.1 Introduction

We present in this chapter several results and methods of interest. The main result, as we have stated it before, is the proof that the domain $\rho < 1$ coincides with the stability domain for a class of scheduling policies. This result can be rephrased as the stability domain of a network s_1, \dots, s_N is the product of the stability domain of its stations:

$$\mathcal{D}_{f_\infty, \dots, f_N} = \mathcal{D}_{f_\infty} \times \mathcal{D}_{f_\epsilon} \dots \times \mathcal{D}_{f_N} \quad (3.1)$$

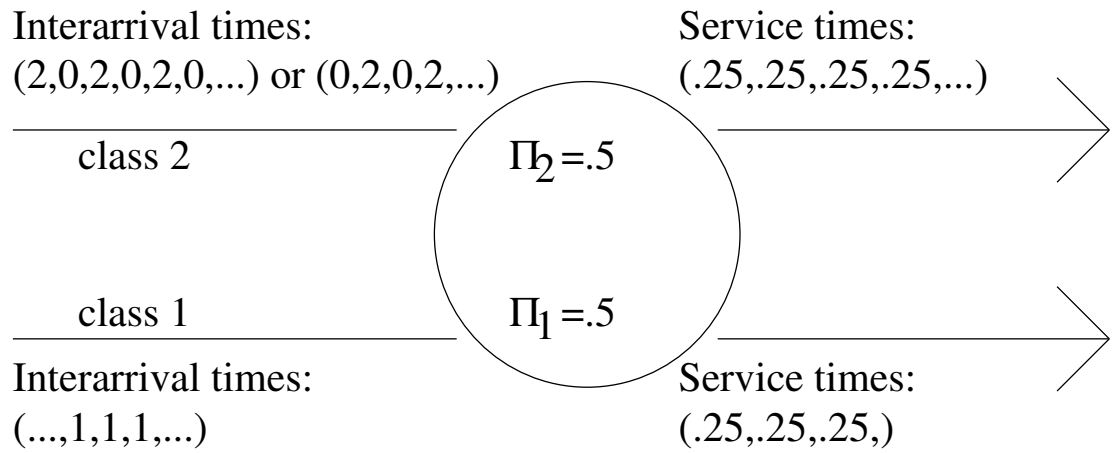
We are indeed proving a separation principle for the network: each station can be studied from the point of view of stability independantly of the others.

3.2 Motivation for rate stability

The rate stability is as strong a result as we can expect from assumptions as light as the stationary and ergodic requirement. Indeed, we present here a very simple network that does not converge to a steady-state distribution.

3.2.1 Counterexample

We give now an example of a very simple network that does not converge to a stationary distribution. It is a 2 classes one station network. Customers from class 1 arrive to the network according to the deterministic arrival process with interarrival time equal to 1. They also require a deterministic service time equal to 0.25. Customers from class 2 arrive according to the arrival process defined as follow: the interarrival times are a periodic sequence, namely $\dots, 2, 0, 2, 0, 2, 0, \dots$. The 0th customer arrives a time 0, and the interarrival time between the 0th and the first customer is 2 or 0, each with a probability $1/4$. Customers from class 2 require a service time, also deterministic with duration 0.25.



A stable two-class network in which customer from class 1 do not have a steady-state stationary waiting time distribution.

Figure 3.1: Non Stationary Network

We can check that the stability condition is satisfied, each one of the two flows brings in 0.25 units of work per unit of time, hence $\rho = .5$.

Now, consider the waiting time of a customer of class 1. Every other customer from class 1 arrives in an empty system, hence its waiting time is 0.

$$W_{2n+1}^1 = 0 \tag{3.2}$$

Now, all the customers $2n$ from class 1 arrive at the same time as 2 customers from class 2, either the customers $2n$ and $2n + 1$ or $2n - 1$ and $2n$. Then, these customers compete for the same resource. Since the flow of customers of class 1 is equal to the flow of customers of class 2, the station tosses a coin with equal probability to buffer 1 or buffer 2. Hence, the waiting time of customer $2n$ from class 1 is:

$$W_{2n}^1 = \begin{cases} 0 & \text{w.p. } 1/2 \\ 0.25 & \text{w.p. } 1/4 \\ 0.5 & \text{w.p. } 1/4 \end{cases} \quad (3.3)$$

As we can see, even though both input sequences and the coin toss are stationary and ergodic, the waiting time distribution of a customer from class 1 is not. Hence the rate stability is a reasonable goal, since convergence to a stationary regime is out of reach.

We now focus on the proof of the rate stability of the random multiplexing policy.

3.3 Rate stability

On the way to proving our main rate-stability result, we will also obtain some other interesting intermediate results. For instance, we identify a scheduling algorithm which is idling, and still stabilizes the system. Such an algorithm is interesting, since it contradicts the intuition that idling the system can only hurt the system.

The method of the proof deserves some attention too, since it illustrates the separation principle we highlighted in this chapter's introduction. The separation principle is dual: we will see that we can consider only one route in the system, and analyse it without taking into account the statistics of the other routes in the system. This would be a transverse separation of the system. We will also see that we can consider one station in the system, and study its stability independently of the other ones.

This type of proof could thus be extended to some other systems, in which the nodes could be more general than multiclass servers. Any system which exhibits such a separation property could be studied in the same way. The idea of the proof is to extract one route of interest, and to couple this route with a system whose properties we know. This proof would thus extend to all systems where some randomness is introduced so as to allow us to separate the different stations.

In order to prove Theorem 2.3.1, we need some preliminary results. We will first consider a tandem network; this network having a simple topology, we can deduce its stability from the arrival and service processes. We will then compare this network and a new tandem network with forced idle periods in the next subsection. By bounding the difference in the departure processes of the idling tandem, and the non-idling tandem network, we will obtain stability of the tandem network with forced idle periods. Eventually, we will compare the departure of this idling tandem network to our original system. By defining some kind of coupling between these two networks, we will be able to prove the stability of our general network. This outlines the proof of Theorem 2.3.1, and motivates the next paragraphs.

3.4 A particular case of tandem network

The point of this section is to analyze a first building block in our proof, namely a very simple tandem network. Such a tandem network will later be linked or compared or coupled to a single route that we will extract from our original network: this is the separation principle we mentioned already. Keeping this purpose in mind will explain some of the notation of this section.

For instance, we consider a tandem network with v stations. We use v instead of S as in the model description because v will correspond to the number of visits to the v buffers a customer from a given route makes. A single route r in the original system visits v_r buffers, each one corresponding to a *station* in the tandem network we present here.

A tandem network is the same as a pipeline network: customers go through a line of stations, always in the same order. Customer j is defined as a $v + 1$ -tuple $(\tau_j, \sigma_j^i, 1 \leq i \leq v)$ corresponding to the interarrival time and the service requested by customer j at each station i .

The sequence indexed by j is supposed to be stationary and ergodic with respect to the corresponding shift. Denote $\lambda_1 = E[\tau_1]^{-1}$. Station $s \in \mathcal{V} = \{1, \dots, v\}$ serves the customers in the following fashion:

Each station s is given a number $p_s \geq 1$ and a sequence $(\pi_i^s, \bar{\sigma}_i^s), i \in \{1, \dots, p_s\}$ satisfying:

- (i) $0 \leq \pi_i^s \leq 1, \quad \forall i \in \{1, \dots, p_s\}$
- (ii) $\sum_{i=1}^{p_s} \pi_i^s = 1$

$$(iii) \quad \bar{\sigma}_i^s \in [0, \infty), \quad \forall i \in \{1, \dots, p_s\} \quad (3.4)$$

Thus, the π_i^s 's define some probability distribution, and the $\bar{\sigma}_i^s$'s correspond to some fixed mean service time. The term mean and the bar notation should not distract from the fact that those quantities are deterministic. They will later correspond to some means when we link this system with our original model. Also the π_j^s will correspond to the coin toss distributions at every station in the original system.

After the service completion of a customer, the server s throws an i.i.d. coin equal to i with probability π_i^s .

- If $i = 1$ it serves the next customer waiting, or waits until the arrival of the next customer which then enters service immediately.
- Otherwise, if $i > 1$, it idles for a period of time $\bar{\sigma}_i^s$.

Since the coin we throw is i.i.d., this system is equivalent (in the 'waiting time of a customer' sense) to serving the customer j a total service time of

$$\theta_j^s = \sigma_j^s + \sum_{i=2}^{p_s} \mu_j^{(i,s)} \bar{\sigma}_i^s, \quad (3.5)$$

where $(\mu_j^{(i,s)})_{j \in \{1,2,\dots\}}$ is an i.i.d. sequence of geometric random variable with parameters π_i^s (in the case $p_s = 1$, the sum in 3.5 is of course equal to zero).

We say equivalent 'in the waiting time of a customer' sense, since the departure process is different when considering a tandem with service times θ_j^s : a customer leaves earlier in our tandem network. Yet the waiting time of a customer is the same.

$\mu_j^{(i,s)}$ for $i > 1$ is the number of times the coin is going to point at i before pointing at 1 after completing the service of customer j at station s . Hence its mean is $\frac{\pi_i^s}{\pi_1^s}$. We can define

$$\lambda_i^s = \frac{\pi_i^s}{\pi_1^s} \lambda_1 \text{ for } 1 < i \leq p_s. \quad (3.6)$$

Due to the independence and the identical distribution of the $\mu_j^{(i,s)}$ with respect to j , the total service time θ_j^s defined in 3.5 as the sum of two stationary and ergodic sequences with respect to the j -index shift, is hence another stationary and ergodic sequence.

By a classical result [53], this system is stable if and only if, for all $s \in \{1, \dots, S\}$:

$$\begin{aligned} \lambda_1 \left[E[\sigma_1^s] + \sum_{i=2}^{p_s} E[\mu_1^{(i,s)}] \bar{\sigma}_i^s \right] &< 1 \text{ or} \\ \lambda_1 \left[E[\sigma_1^s] + \sum_{i=2}^{p_s} \frac{\lambda_i}{\lambda_1} \bar{\sigma}_i^s \right] &< 1 \text{ or} \\ \sum_{i=1}^{p_s} \lambda_i \bar{\sigma}_i^s &< 1 \end{aligned} \tag{3.7}$$

where $\bar{\sigma}_1^s = E[\sigma_1^s]$. It is stable in the strong sense that the waiting time distribution converges to a stationary distribution in the long run, which is almost surely finite. This is Loyne's result.

The condition 3.7 is written so as to look like the stability condition: this is why we assume from now on that it is satisfied, and that our tandem network is thus strongly stable.

In this section, we defined a tandem network so that it resembles a single route in our general network. We will define in the next section another network that is a tighter fit to a single route in our original network.

3.4.1 Tandem network with forced idle periods

We consider now a slightly different system. The customer input stream and the tandem structure are the same as in the previous system. The difference lies in the serving policy.

Now, the server $s \in \mathcal{V}$ tosses the same coin, pointing at i with probability π_i^s . If the outcome of the toss is $i = 1$ and there is a customer waiting to be served, this customer goes into service. If there is no customer waiting and $i = 1$, then the server idles for a time $\bar{\sigma}_1^s$. If the outcome of the toss is $i > 1$ (whenever this is possible), then the server idles for a time $\bar{\sigma}_i^s$.

The difference between this system and the one in the previous section is that the server does not wait anymore for the next customer when pointing at 1, but instead idles (or serves some ghost customer with the mean service time of its class). It is a non-work-conserving system. The server adds a load to the system. A customer entering a station will *always* be delayed by some residual idling period, even though the system might be empty of other real customers. In order to differentiate this idling tandem from the tandem network in the previous section, we will denote the quantities corresponding to the idling tandem with a tilde. And reciprocally, we will call the idling tandem the 'tilde system'.

Coupling We *couple* the coin tosses of the tandem network and the tilde system such that whenever a given customer finishes service, the subsequent coin tosses in both system are equivalent, until the coin points again at buffer 1. We can impose this condition, since the coin tosses are distributed in the same manner, and are independent from the other random variables.

For instance, customer 1 arrives in station 1: it goes into service in the tandem network, it is delayed in the tilde system. After its departure, both systems' coin toss sequences coincide until they again point at buffer 1.

Stability Claim As far as the stability of the tilde system goes, the condition 3.7 is still necessary and sufficient. An intuitive way to see this is to notice that only when buffer 1 is empty does the tilde system idle. Hence, in the case of unstability, customers would be stuck in buffer 1, and the new tilde system would behave exactly as the particular tandem of the previous section. We need some notation to explain this formally.

Define by a_j^s and d_j^s the arrival time and departure time of customer j at station s in the tandem of the previous subsection, and by \tilde{a}_j^s and \tilde{d}_j^s the corresponding values in our new system.

We will proceed by induction.

Initial step We have the obvious relations:

$$\begin{aligned} a_j^1 &= \tilde{a}_j^1 \\ a_j^{s+1} &= d_j^s \\ \tilde{a}_j^{s+1} &= \tilde{d}_j^s \end{aligned}$$

for $s = 1, \dots, v - 1$. Now, consider a busy period of station 1 from the tandem of the previous section. By busy period, we mean that whenever station 1 of the system without tilde calls on a customer (case $i = 1$), a customer is present and waiting for service.

The first customer of a busy period enters service immediately, in the tilde-less system. In the tilde system, the corresponding customer (with the same index) is delayed some amount of time. This amount of time is less than

$$\bar{\sigma}_1^1 + \sum_{i=2}^{p_1} \mu^{(i,1)} \bar{\sigma}_i^1 \tag{3.8}$$

where $\mu^{(i,1)}$ are some geometric random variables independent of everything else.

Recall that in the tilde system, a customer arriving in an empty system still has to wait until the coin toss points at buffer 1; whenever the coin toss points at another buffer $i \neq 1$, the customer is delayed $\bar{\sigma}_i^1$.

The $\mu^{(i,1)}$ correspond to the number of times the tilde system points at buffer i between the last time it points at buffer 1 prior to the arrival of the first customer of the busy period, and the first time it points at buffer 1 after the arrival of the first customer of the busy period. Namely, $\mu^{(i,1)}$ are geometric with parameter π_i^1 .

The term $\bar{\sigma}_1^1$ correspond to the situation in which the first customer of the busy period arrives while the tilde system is idling after pointing at an empty buffer 1.

The subsequent customers (corresponding to those of the busy period of the tandem system) will see the same extra delay as the first customer. Indeed, they have the same arrivals in both systems, hence a busy period of the tandem network is also a busy period of the tilde system. During a busy period of the tandem network, the coin toss sequences are identical, by the coupling we imposed.

For each busy period, the delay will differ, yet it is always of the form 3.8. Hence, for all indices of the customers, the difference between the hat system and the tilde system is bounded by some random variable in the form of 3.8 which is nice enough so as to preserve the stability:

$$\frac{\tilde{d}_j^1 - d_j^1}{f(j)} \xrightarrow{j \rightarrow \infty} 0 \text{ a.s.} \quad (3.9)$$

for any function $f(\cdot)$ increasing to ∞ .

Induction step In the same manner, we can bound the difference between the departure times of the tandem system and the tilde system for the subsequent stations by induction.

Assume the arrival processes at station s are such that:

$$\tilde{a}_j^s - a_j^s < \delta_j^s \quad (3.10)$$

where δ_j^s is defined to be a compound sum of random variables of the form 3.8. δ_j^s is the *delay* between the two systems.

Consider a busy period of the tilde system. The first customer f of a busy period of station s of the tilde system sees a delay which is bounded by:

$$\delta_f^{s+1} = \delta_f^s + \bar{\sigma}_1^s + \sum_{i=2}^{p_s} \mu^{(i,s)} \bar{\sigma}_i^s \quad (3.11)$$

The subsequent customers of the busy period of station s in the tilde system do not see any increase in their delay compared to the tandem network, since the coin toss sequence are coupled.

Thus for all indices j , we can define a δ_j^{s+1} such that:

$$\tilde{d}_j^s - d_j^s < \delta_j^{s+1} \quad (3.12)$$

where δ_j^{s+1} is of the form 3.8.

This complete the induction proof:

$$\forall s \in \mathcal{V}, \frac{\tilde{d}_j^s - d_j^s}{f(j)} \rightarrow_{j \rightarrow \infty} 0 \text{ a.s.} \quad (3.13)$$

Equation 3.13 gives a strong stability result: the system with no tilde converges to some stationary distribution (cf. [53]), and the tilde system differs by only a tiny amount. For instance, the tilde system is more than rate stable. Rate stability states that the input rate equals the output rate. This would be given by taking the function f equals to the function identity, since the system with no tilde is obviously rate stable. Since we can choose any function increasing to ∞ , we have a much stronger result.

This completes the proof of our result for the tilde system: it is stable whenever the stability condition 3.7 is satisfied.

3.5 Proof of Theorem 2.3.1

We now consider one route $r \in \mathcal{R}$ in our original system. r visits v buffers, $(b_1, b_2, \dots, b_v) \in \mathcal{B}^v$. With no loss of generality, we can assume that $b_i = i$ and that the route r visits buffers $(1, 2, \dots, v)$.

As expected, we define the corresponding tilde system: it is a v stations tandem network, fed with the same input as the route r . For the k th station of the tilde system, we look at

the k th buffer in the sequence r . b_k belongs to station $s \triangleq s_k$. Station s receives customers from p_s buffers. Hence p_s is the cardinality of the set:

$$p_s = |\{b : s_b = s\}| \quad (3.14)$$

Consider g to be a bijection from $\{1, \dots, p_s\}$ onto $\{b : s_b = s_k = s\}$ such that $g(1) = b_k = k$. The parameters of the tilde system defined in equation 3.4 (i),(ii),(iii) are:

$$\begin{aligned} \tilde{p}_s &= p_s \\ \tilde{\pi}_i^k &= \pi_{g(i)}^s \\ \tilde{\sigma}_i^k &= E[\sigma_1^{g(i)}] \end{aligned} \quad (3.15)$$

with the obvious convention that notations with a tilde refer to the tilde system. It is obvious that they satisfy equation 3.4 (i),(ii),(iii). Also, note that according to these parameters, the flow in the tilde system is $\tilde{\lambda}_1 \triangleq \lambda_1 = \lambda^r$ and that:

$$\tilde{\lambda}_i^k \triangleq \tilde{\lambda}_i^k = \frac{\tilde{\pi}_i^k}{\tilde{\pi}_1^k} \tilde{\lambda}_1 = \frac{\pi_{g(i)}^s}{\pi_g^s(1)} \lambda^r = \lambda_{g(i)} \quad (3.16)$$

Coupling Here again, we couple the coin toss sequences of the tilde system with the system to which we wish to make a comparison. We fix the coin toss sequence of the tilde system such that it coincides with the coin toss sequence of the original network, namely that *the coin toss of the tilde system after the service completion of the j th customer in buffer 1 of the tilde system points at i whenever the coin toss of the original system points at $g(i)$ after the service completion of the j th customer in buffer k in the original system.*

Such a coupling is possible since the distribution is defined to achieve such a coupling. Note that the coin tosses of the tilde system at two different stations are not independent anymore from one sequence to another. Yet, the outcome of the coin tosses within the sequence at one fixed station are still i.i.d., which is all we need to apply our result.

More definitions We can define the quantity a_j^b and d_j^b to be the arrival time and the departure time of the j th customer from buffer b . We recycle the notations since there is no risk of confusion. The tandem network with no idling was an intermediate step to which we will not come back to. In this section, whenever there is no tilde on a notation, we refer to a quantity of the original network.

We note that the stability of the tilde system is ensured by 3.7, which in turn is implied by the stability condition on our original network 2.7 due to the remark 3.16. This is obvious since we chose the parameters of the tilde system such that they coincide with a station of the original system. Also note the tilde system is an idling system, whereas the real system is work conserving.

We will prove a first lemma:

Lemma 3.5.1 *For a customer, say j , arriving in an empty real (resp. tilde) buffer b ,*

$$\frac{a_j^b - d_j^b}{j} \rightarrow 0 \text{ a.s. } \left(\text{resp. } \frac{\tilde{a}_j^b - \tilde{d}_j^b}{j} \rightarrow 0 \text{ a.s. } \right) \quad (3.17)$$

The proof is essentially the same for the real or the tilde system. We give it for the real system. We leave it to the reader to add bars and tildes for the tilde system.

First, define for customer j entering the buffer b at station s_b , the following indices: (j, b) is the j^{th} customer from buffer b ; for β a buffer feeding customers to station s_b , i.e. such that $s_b = s_\beta$, denote by $l_\beta^{(j,b)}$ (and respectively $h_\beta^{(j,b)}$) the lower (respectively the higher) indice of the set of customers from buffer β which enter or complete service after (j, b) arrives and before (j, b) goes into service.

Thus, (j, b) is directly affected by all the customers (i, β) for all $i \in \{l_\beta^{(j,b)}, \dots, h_\beta^{(j,b)}\}$ and all β such that $s_\beta = s_b$.

The time spent in the system by customer j entering in an empty buffer b is bounded above by a term of the form:

$$w^{(j,b)} = \sigma_j^b + \sum_{\beta \in s_b} \sum_{k=l_\beta^{(j,b)}}^{h_\beta^{(j,b)}} \sigma_k^\beta \quad (3.18)$$

corresponding to the service of all customers scheduled ahead of (j, b) after (j, b) 's arrival in an empty buffer b . We denote this quantity $w^{(j,b)}$ since it is a bound on the waiting time of customer j at buffer b . It is an upper bound because (j, b) only sees the residual life of the last customer that entered service prior to (j, b) 's arrival.

The difference $h_\beta^{(j,b)} - l_\beta^{(j,b)}$ is bounded above by a geometric random variable with parameter $\frac{\pi_\beta^s}{\pi_b^s}$ independent of j , due to the independence of the coin tosses from the input process and the distribution of the coin toss according to the π_β^s for $\beta \in s$. $(h_\beta^{(j,b)} - l_\beta^{(j,b)})$ is actually

equal to a geometric random variable whenever all the buffers pointed at by the coin toss are non-empty at the time of the coin toss. If customers are absent at the time of the coin toss, then the toss is not accounted for in the sum, and the bound is a strict upper bound.

We need to show that $w^{(j,b)}$ defined in 3.18 is $o(j)$. It is a two step process:

1. first assume that $h_\beta^{(j,b)} - l_\beta^{(j,b)}$ grows to ∞ for some subsequence. Then by Birkhoff's ergodic theorem, (a description of which can be found in [27]),

$$\frac{\sum_{k=l_\beta^{(j,b)}}^{h_\beta^{(j,b)}} \sigma_k^\beta}{h_\beta^{(j,b)} - l_\beta^{(j,b)}} \rightarrow \bar{\sigma}^\beta \text{ a.s.} \quad (3.19)$$

and, together with

$$\frac{h_\beta^{(j,b)} - l_\beta^{(j,b)}}{j} \rightarrow 0 \text{ a.s} \quad (3.20)$$

due to the bound in distribution on $(h - l)_\beta^{(j,b)}$, it yields the result.

2. Now for the bounded subsequences, it is obvious, also by Birkoff's ergodic theorem, that

$$\frac{w^{(j,b)}}{j} \rightarrow 0 \text{ as } j \rightarrow \infty \quad (3.21)$$

In any case, we obtain the result. ■

We now want to establish the following lemma

Lemma 3.5.2

$$\frac{d_j^b - \tilde{d}_j^b}{j} \rightarrow 0 \text{ a.s. as } j \rightarrow \infty. \quad (3.22)$$

We proceed by induction.

Consider buffer 1 of the real system (as opposed to the tilde system). Recall that we consider route $r = (1, 2, \dots, v)$. We define a busy period of this buffer as a set of consecutive indices of customers from buffer 1 for which whenever the station s_1 decides, according to the coin toss, to pick a customer from buffer 1, buffer 1 is not empty. For the j th customer of buffer 1 being served at station s_1 , namely $(j, 1)$, denote by f_j the first indice of its busy period ($f_j \leq j$).

Two buffer case We will assume, in order to improve the readability of the proof that station s_1 has only two buffers. The case when it has one buffer is trivial (both the real and the tilde system proceeds exactly according to the same fashion during a busy period), and the case when there are more than two buffers is treated in exactly the same fashion as the two buffers case.

Denote by b_o the other buffer of station s_1 . Denote by \mathcal{B}_j the indices of the customers from buffer b_o actually served by the real system between $d_{f_j}^1$ and d_j^1 . \mathcal{B}_j denotes the customers from b_o served during the busy period of s_1 leading to customer j . The cardinality of \mathcal{B}_j , satisfies the following relationship, since the busy period starts at customer f_j and not customer 1, and due to the distribution of the coin tosses:

$$\frac{|\mathcal{B}_j|}{j} \leq \frac{\pi_{b_o}^{s_1}}{\pi_1^{s_1}} \text{ a.s.} \quad (3.23)$$

By construction, we know that the coin toss of the tilde system and the real system follow the same sequence from the departure of customer f_j on. This construction is possible since they follow the same i.i.d. distribution. Also, there is no possible overlap of the indices, hence no possibility of dependent or contradictory coin tosses. In one word, we couple the coin tosses of the two system from f_j to j . If the coin toss points to an empty buffer in the real system, we still consider this toss in the tilde system (and idle the tilde system accordingly).

The tilde system may not have a busy period of its own between f_j and j . Denote by I the number of times the tilde system idles the server for buffer 1 being empty and the coin pointing at buffer 1 during the busy period $\{f_j, \dots, j\}$.

Of course, if buffer 1 of the tilde system is having a busy period of its own, then $I = 0$. Otherwise, I times will the tilde server idle when the real server actually serves a customer.

It is important to remark that for the tilde system, by its stability,

$$\tilde{d}_j^k \sim a_j^k \text{ a.s. as } j \rightarrow \infty. \quad (3.24)$$

Also, since the customer f_j in the real system finds an empty buffer, it also satisfies

$$d_{f_j}^k \sim a_{f_j}^k \text{ a.s. as } j \rightarrow \infty. \quad (3.25)$$

by lemma 3.17

Putting 3.24 and 3.25 together yields, thanks to $a_j^1 = \tilde{a}_j^1$:

$$d_{f_j}^1 \sim \tilde{d}_{f_j}^1 \text{ a.s. as } k \rightarrow \infty. \quad (3.26)$$

During the busy period of buffer 1 in the original system, i.e. in between customers f_j and j of buffer 1, four situations might arise:

1. the coin toss points at buffer 1, which is non empty in both systems.
2. the coin toss points at buffer 1, and the tilde system is empty. This case yields a difference in service time of the form $\sigma_{(\cdot)}^1 - \bar{\sigma}^1$.
3. the coin toss points at buffer b_o and the original system is empty. This delays the tilde system an extra $\bar{\sigma}^2$.
4. the coin toss points at buffer b_o and the original system is non-empty. This induces a delay difference of $\sigma_{(\cdot)}^2 - \bar{\sigma}^2$.

Buffer 1 of the original system and buffer b_o of the tilde system are never empty.

The following array illustrates one possible parallel behavior of the real and the tilde systems from customer f_j on. In this array, we write the outcome of the coin toss, which is coupled for the two systems, and the action taken by each system for the set of customers $\mathcal{B}_j = \{f_j, f_j + 1, \dots, j\}$ in buffer 1 of the original system. A dot indicates a coin toss at an empty buffer; if the coin toss points at a non-empty buffer, then the service time of the customer called into service is written.

A possible trajectory

| | | | | | | | | | |
|--------------|------------------|--------------------|----------------------|----------------------|--------------------|-----|----------------------|----------------------|------------------|
| real system | $\sigma_{f_j}^1$ | $\sigma_{f_j+1}^1$ | $\sigma_k^{b_o}$ | . | $\sigma_{f_j+2}^1$ | ... | $\sigma_{l-1}^{b_o}$ | $\sigma_l^{b_o}$ | σ_j^1 |
| coin toss | 1 | 1 | b_o | b_o | 1 | ... | b_o | b_o | 1 |
| tilde system | $\sigma_{f_j}^1$ | $\bar{\sigma}^1$ | $\bar{\sigma}^{b_o}$ | $\bar{\sigma}^{b_o}$ | $\sigma_{f_j+1}^1$ | ... | $\bar{\sigma}^{b_o}$ | $\bar{\sigma}^{b_o}$ | σ_{j-I}^1 |

Table 3.1: A possible sample path

This illustration helps us write an equation linking d_j^1 and \tilde{d}_{j-I}^1 as a function of the time the busy period started. Indeed:

$$d_j^1 - \tilde{d}_{j-I}^1 \leq |d_{f_j}^1 - \tilde{d}_{f_j}^1| + \left| \sum_{i \in \mathcal{B}_j} \sigma_i^{b_o} - \bar{\sigma}_i^{b_o} \right| + \left| \sum_{i=j-I+1}^j \sigma_i^1 - \bar{\sigma}_i^1 \right| \quad (3.27)$$

After the start of a busy period of station 1 of the real system, we can identify the differences between the two systems. Whenever the real system points to an empty buffer (b_o since we consider a busy period of buffer 1), the tilde system idles an extra delay, which is omitted in the upper bound. Whenever the tilde system serves a ghost customer from buffer b_o (or equivalently idles the mean service time of buffer b_o) and the real system serves a real customer, the difference is accounted in the second term of the R.H.S. Whenever the tilde system serves a ghost customer from buffer 1 and the real system serves a real customer from buffer 1, the difference is accounted in the last term of the R.H.S. Those are all the possible differences, which completes the explanation of equation 3.27. All these cases are illustrated in the previous array.

Now, it is obvious that, when divided by j , the last two terms of the R.H.S will vanish, by Birkhoff's ergodic theorem. Indeed, consider one of these terms:

$$\frac{\sum_{i \in \mathcal{B}_j} \sigma_i^{b_o} - \bar{\sigma}^{b_o}}{j} = \frac{\sum_{i \in \mathcal{B}_j} \sigma_i^{b_o} - \bar{\sigma}^{b_o}}{|\mathcal{B}_j|} \frac{|\mathcal{B}_j|}{j} \quad (3.28)$$

If $|\mathcal{B}_j| \rightarrow \infty$, or for any subsequence thereof going to ∞ , then the first term of the R.H.S. of equation 3.28 goes to 0 and the second term is bounded by $\frac{\pi_{b_o}^{s_1}}{\pi_1}$ almost surely by equation 3.23, hence the L.H.S. goes to 0. If $|\mathcal{B}_j|$ is bounded, then we have a bounded number of terms in the sum, and dividing by j goes to 0 almost surely.

The first term of the R.H.S. of equation 3.27 vanishes due to equation 3.26. Hence

$$a_j^1 \leq d_j^1 \leq \tilde{d}_{j-I}^1 \leq \tilde{d}_j^1 \sim \tilde{a}_j^1 = a_j^1 \quad (3.29)$$

This proves that the departure rate from buffer 1 in the real system is the same as in the tilde system.

Now, we can see that the induction step will be exactly the same, with the only difference that $\tilde{a}_j^k \sim a_j^k$ instead of $\tilde{a}_j^1 = a_j^1$. This concludes the proof of the lemma

This concludes the proof of theorem 1, since the route r was generic and the tilde system is stable. ■

Note that we can only prove rate stability within this set up, since by dividing equation 3.27 by $o(j)$ instead of j , it would not satisfy Birkhoff's ergodic theorem hypothesis anymore, and the upper bound would not necessarily go to 0.

3.6 An example of stable idling policy

The main part of the proof is to obtain equation 3.27. Equation 3.27 states that the original system cannot stray away from the tilde system by more than a $o(j)$ quantity. Hence, any 'original' system satisfying equation 3.27 would also be stable, and not only the model defined in the previous chapter.

Consider for instance a model in which the scheduling policy is as follows: for each station s , we are given the same probability distribution π_b^s for $b \in s$. The coin toss proceeds exactly in the same way as in our original model, except that whenever it points at an empty buffer b_e , instead of pointing again at another one if there is a non-empty buffer at station s , or waiting for the next customer to arrive if all buffers of station s are empty, it idles the system a time $\bar{\sigma}^{b_e}$.

If we look carefully at how we obtained equation 3.27, we can see that this system mimics more closely the behavior of the tilde system. The bound between this idling system and the tilde system is tighter.

Thus this quite naturally idling system is also rate-stable. Of course, by removing the non-idling condition, we enlarge the set of possible scheduling policies, hence we enlarge also the possible stability domain for the network.

3.7 Conclusion

We proved that for a general network, the random multiplexing queueing policy ensures stability, whether it is idling or not, and that we can loosely couple our network with a set of modified tandem queues. Hence we can approximate the asymptotic position of a customer by looking at its equivalent in the 'tilde' system. An interesting fact is the separability property of the RM policy, since each buffer really functions like it is in some isolation from the rest of the system.

The main result is that for any network topology, the stability domain given by equation 2.7 is achievable. Also we have defined a distributed policy that requires only the local knowledge of the local flow and the local mean time spent by customers in the station. Since it is an asymptotic result, in a practical framework, the mean time of customer from a given

class can be estimated on the run. The flow can also be approximated on the run by counting the customers. The complexity of the policy is that of tossing an i.i.d. coin, which is independent of the size of the network.

To phrase this result in yet another way, we have defined a scheduling policy that would beat any adversarial model. FCFS was conjectured for a long time to be stable, until a counter example was found. For most classical scheduling policies, there is an example of unstability. For our scheduling policy, we proved the impossibility of constructing such a counter example.

We will show in the sequel that a scalable scheduling policy just receiving the information in the form of a count of the customers from different types arriving at the node would achieve the maximum theoretical stability region as well. This would answer all doubts about implementing effectively our scheduling policy.

We discussed a little bit the quality of the stability. We said that the tilde system was more strongly stable than just rate stable. We also said that our scheduling policy performed better than a random multiplexing idling policy that was closer to the tilde system. Hence, there is a strong indication that we can reach a stronger stability result. This is the object of the next chapter.

Chapter 4

Strong stability in Markovian networks

4.1 Introduction

We have proved in the previous chapter that the random multiplexing policy would prevent the appearance of blocking patterns and oscillatory modes of amplitude proportional to the number of customers entering the system or to time. Any such phenomenon could at most grow sublinearly. Yet, the number of customers in the system could be proportional to \sqrt{j} or $\log(j)$, and grow to infinity on a set of sample paths of positive measure. This would somehow contradict the notion that the system is stable.

We cannot imagine stability as a fixed strict upper bound on the number of customers in the system. Such a constraint is unrealistic. We have to allow the size of the system to get as big as can be. One reason would be to be consistent with the assumption that all the buffers have infinite capacity. A better reason, for instance, is that we know the distribution of the number of customers in a M/M/1 single class queue, which is geometric, and is not bounded. Our model has to encompass this simple case, even though we would like to be able to say that the probability that the number of customers increases to infinity is null.

In this chapter, we will give a bound on the waiting time of customers in the system. This bound has to be in distribution, as we just mentioned. The distribution has to be honest, in the sense that the measure of ∞ has to be zero. The number of customers has to be

asymptotically finite with probability one. This is a strong stability result, as opposed to the rate stability result we attained earlier. We will identify a coupled system that converges to a steady-state distribution, and has longer waiting time than our original system. Hence, the limiting distribution of the waiting time in the coupled system will be an upper bound in distribution for our original system.

This result of course includes the result from the previous chapter, and can be obtained independently.

We restrict ourself to a particular case: we consider the arrival sequences and the service sequences to be composed of identically and independently distributed variables.

This assumption is more restrictive than the stationarity and ergodicity we imposed earlier. Yet, it still is a reasonable assumption in many applications. Intuitively, it means that the service request of a customer does not influence the service request of the next customer from the same class at the same station, or that the process generating external arrival into the network regenerates itself after each external arrival. Also, the result we attained earlier dealt only with averages: it seems natural to impose more conditions to reach a stronger result, namely a convergence to some distribution of the steady state of the system.

We intend to prove two results, first a bound on the waiting time of any customer at any station by some random variable finite almost surely. Then, we will deduce from this result the convergence to a stationary regime of our network.

In order to build some intuition regarding the proof, we consider one more time the simple Lu-Kumar network in the next section, and prove the existence of a distributional bound on the waiting time of a customer.

Then we extend this proof to the general case in the following section. Eventually, we prove the convergence to equilibrium in the last section of this chapter.

4.2 I.I.D Lu-Kumar network

4.2.1 Introduction

In this section, we consider two sequences, a sequence of interarrival times $(\tau_j, j \in \mathcal{Z})$ and a sequence of service times $(\sigma_j^1, \dots, \sigma_j^4, j \in \mathcal{Z})$, each composed of i.i.d random variables.

Customers arriving according the (τ_j) interarrival sequence get service first at station 1, then at station 2, then again at station 2, and back at station 1, according to the σ_j^i sequences. As we have seen before, this is a 2 stations 4 classes network.

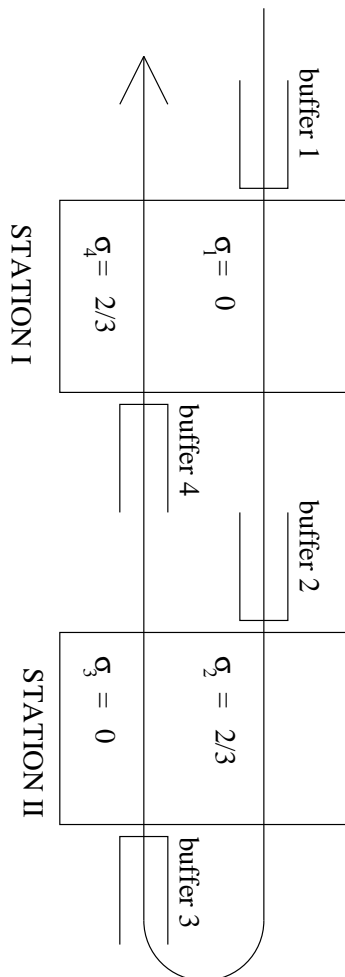


Figure 4.1: Lu-Kumar network

We now state the result we wish to obtain in this particular case:

Theorem 4.2.1 *The random multiplexing policy is strongly stable in the Lu-Kumar network, in the sense that the waiting time is bounded in the limit by a stationary almost surely finite random variable.*

We want to prove a bound on the waiting time of any customer by induction on the class. We can number the buffers in the visiting order, and associate a class with a buffer.

Now, we construct a bound on the waiting time of class 1 customers, and deduced some property of the outgoing flow of customers of class 1 from station 1: this is the arrival flow into class 2.

The iteration step is to prove that the property of the incoming flow is enough to obtain a bound on the waiting time for customers in class 2 and the same property for the outgoing flow. This in turn would prove our result for class 3 and 4.

First, we need to define the property that needs to be satisfied. We introduce some notations:

By A_j^{i+} and A_j^{i-} denote respectively the stationary processes bounding the arrival process of customers to buffer i A_j^i above and below, if such processes exist. Namely, on a given sample path, the arrival time of customer j in buffer i satisfies:

$$A_j^{i-} \leq A_j^i \leq A_j^{i+} \quad (4.1)$$

By W_j^i denote the waiting time of customer j from class i , that is the time that customer j spend in buffer i .

Property

The customer of class i whose arrival process to buffer i is bounded above and below by two given processes converging to two stationary processes has a waiting time process bounded above by a process converging to a stationary and almost surely finite process. Furthermore, the departure process from class i is also bounded above and below by two given processes converging to two stationary processes.

To put this in mathematical terms: assuming there exist two processes A^{i+} and A^{i-} converging to steady state as $i \rightarrow \infty$ and satisfying the inequality 4.1, then we can construct a distribution bound on the waiting time W_j^i which we denote W_j^{i+} , such that:

$$W_j^i < W_j^{i+} + (A_j^{i+} - A_j^{i-}) \quad (4.2)$$

Furthermore, we can also construct two processes A^{i+1-} and A^{i+1+} that converge to a steady state distribution such that the departure from buffer i , namely A_j^{i+1} satisfies the inequality 4.1 as well.

It is clear from the definition of this property that if it is satisfied at step 1, then it implies it is satisfied at step 2, 3 and 4.

4.2.2 Initial step

We check that it is true at step 1.

The arrival process at buffer 1 is the external arrival: since the interarrival times are i.i.d random variables, it is a stationary process, hence we can bound it above and below by itself.

Namely, by defining

$$A_j^{1-} = A_j^1 = A_j^{1+} \quad (4.3)$$

we can check that our property is satisfied.

Now we need to check this implies (i) the same for the departure process from station 1 of customer from class 1 and (ii) the bound on the waiting times. This is the iteration step, and we consider it in the next section.

4.2.3 Iterative step

Lemma 4.2.1 *For all buffers i , we can construct two waiting time sequences W_j^{i+} and W_j^{i-} converging to some steady state such that:*

$$W_j^{i-} - (A_j^{i+} - A_j^{i-}) \leq_{\mathcal{D}} W_j^i \leq_{\mathcal{D}} W_j^{i+} + (A_j^{i+} - A_j^{i-}) \quad (4.4)$$

where $\leq_{\mathcal{D}}$ is a distributional inequality.

Proof:

This proof is made in three steps. We present here the architecture of the proof before explaining each of the steps.

1. We first define a Lower Bound System which has a waiting time \tilde{W}_j^{i-} such that

$$\tilde{W}_j^{i-} \leq W_j^i \text{ a.s.} \quad (4.5)$$

2. We then define an Upper Bound System which has a waiting time \tilde{W}_j^{i+} such that:

$$W_j^i \leq_{\mathcal{D}} \tilde{W}_j^{i+} \quad (4.6)$$

where $\leq_{\mathcal{D}}$ is a bound in distribution.

The sequences $\tilde{W}_j^{i\pm}$ only satisfy the inequalities, but not necessarily the convergence requirements to steady-state.

3. We eventually construct W_j^{i-} and W_j^{i+} satisfying the result of the lemma

We write this proof for the waiting time of a customer from class 2 at station 2; the proof would be exactly the same for a customer from class 3 by substituting class 2 to class 3 and class 3 to class 2. It would exactly be the same also for a customer from class 1 or class 4, by again substituting class 1 for class 2 and class 4 for class 3.

1. Now, consider a customer from class 2 entering its buffer. It is delayed by customers already in the buffer from its class, and by the customers from buffer 3 called ahead of this customer from class 2.

If we assume there are no customers from buffer 3, we get a lower bound on the waiting time. If on the contrary, we assume that there is no customer from buffer 3 missing, then we get an upper bound on the waiting time. To make this intuitive reasoning more explicit, consider the two following G/G/1 systems: the lower bound system LBS is a single server queue receiving customers according to the arrival process A_j^2 and giving these customers a service time σ_j^{2-} which we define to be equal to σ_j^2 . It is the system with no customers ever in buffer 3, hence no delay induced by class 3 on class 2. Since the arrival rate is less than the service rate, the waiting time of such a system, which we denote \tilde{W}_j^{2-} is finite almost surely.

The upper bound system UBS is a single server queue receiving customers according to the arrival process A_j^2 and giving these customers a service time defined as follow: denote by μ_j the number of times the scheduling coin toss points at buffer 3 between customers j and $j + 1$ of buffer 2. Namely, μ_j is an i.i.d. geometric random variable with mean 1. Denote also by χ_k for $1 \leq k \leq \mu_j$ μ_j independent outcomes of the distribution σ_0^3 . Then:

$$\sigma_j^{2+} = \sigma_j^2 + \sum_{k=1}^{\mu_j} \chi_k \quad (4.7)$$

This is another i.i.d. sequence, with mean $E[\sigma_0^2] + E[\sigma_0^3] = \bar{\sigma}^2 + \bar{\sigma}^3$, by the distribution of μ_j and χ_k .

This system is another G/G/1 queue with arrival rate less than the service rate, hence the waiting time of customer j in the UBS is finite almost surely. Denote this waiting time by \tilde{W}_j^{2+} .

It is obvious that:

$$\tilde{W}_j^{2-} \leq W_j^2 \text{ a.s.} \quad (4.8)$$

since arrival processes are the same, and the service times of the LBS are always less. This was the first step of the proof.

2. **Coupling:** We can couple the χ_k and σ_k^3 sequences and the μ_j and coin toss sequence in the following manner:

μ_k corresponds to the number of times the coin toss points at buffer 3 in between two consecutive times pointing at buffer 2, since the coin toss is an i.i.d. Bernoulli sequence of equiprobable outcomes. Thus there is a path with the same probability as the coin toss sequence such that we can map the sequence μ_j to the outcomes of the coin toss immediately after the coin toss that gets customer j into service.

Now, when customer j completes its service, the coin toss sequence might point a number μ_j times to buffer 3, and might or might not admit customers from buffer 3, depending on the fact that these customers are present or not at the time they are being called.

We couple the sequence χ_k with the sequence σ_i^3 such that χ_k is equal to the service time σ_i^3 if this customer i is actually present when the coin toss points at buffer 3 after completing service of customer j , and thus actually receives service. If the coin toss points at an empty buffer 3, then χ_k is taken from the same distribution, independently of the other variables.

Using this coupling, it is easy to see that:

$$W_j^2 \leq_{\mathcal{D}} \tilde{W}_j^{2+} \quad (4.9)$$

where $\leq_{\mathcal{D}}$ is the inequality in distribution, since we constructed a generic outcome of our UBS that bounds W_j^2 .

Note that we make use of the iid assumption in this last inequality: we can couple the χ sequence with the σ^3 sequence only because of the iid assumption. This completes the second step of the proof of the lemma.

3. Define the stationary upper bound system SUBS to be as follow: it is a G/G/1 queue with arrival process A_j^{2+} and service times σ_j^{2+} . Since the arrival process converges to a stationary regime, and the service process is i.i.d., then the output process of such

a system is also converging to stationarity, and so is the waiting time process, which we denote W_j^{2+} .

Symmetrically, define the stationary lower bound system SLBS to be a G/G/1 queue with arrival process A_j^{2-} and service times σ_j^{2-} . The SLBS has a waiting time process W_j^{2-} , which is converging as well to steady state.

Now, equations 4.8 and 4.9 are not the ones we wish for: the waiting times $\tilde{W}_j^{2\pm}$ do not necessarily converge to some steady state distribution. Yet, we can bound \tilde{W}_j^{2+} above and \tilde{W}_j^{2-} below by some random processes having this convergence property.

Namely: consider for instance the UBSsystem. This is a G/G/1 server satisfying its classical stability condition $\rho < 1$. Hence, for any customer in it, say customer 0, we can identify the last busy period preceding the service of this customer.

Denote by $\{-k, -k+1, \dots, 0\}$ the set of indices for the UB G/G/1 system corresponding to the busy period leading to customer 0. Such a busy period is finite with probability one (it is the busy period of a G/G/1 system with arrival rate less than the service rate). The waiting time \tilde{W}_0^{2+} is given by Lindley's equation (a discussion of which can be found in [39]). For a G/G/1 system, Lindley's equation states that:

$$w_{k+1} = \max(0, w_k + u_k) \quad (4.10)$$

with u_k the difference between the service time of customer k minus the interarrival time between k and $k+1$. Of course, during a busy period, all the waiting times are > 0 , and thus the maximum function is not necessary. This is why we can write the first one of the following equations:

$$\begin{aligned} \tilde{W}_0^{2+} &= \left(\sum_{i=-k}^{-1} \sigma_i^{2+} \right) - (A_0^2 - A_{-k}^2) \\ &= \sum_{i=-k}^{-1} \sigma_i^{2+} - (A_0^{2+} - A_{-k}^{2+}) + (A_0^{2+} - A_0^2) - (A_{-k}^{2+} - A_{-k}^2) \\ &\leq \left(W_{-k}^{2+} + \sum_{i=-k}^{-1} \sigma_i^{2+} - (A_0^{2+} - A_{-k}^{2+}) \right) + (A_0^{2+} - A_0^2) \text{ since } 0 \leq W_{-k}^{2+} \\ &\leq \left(\max(0, W_{-k}^{2+} + \sigma_{-k}^{2+} - (A_{-k+1}^{2+} - A_{-k}^{2+})) + \sum_{i=-k+1}^{-1} \sigma_i^{2+} - (A_0^{2+} - A_{-k+1}^{2+}) \right) + (A_0^{2+} - A_0^2) \\ &\leq \left(W_{-k+1}^{2+} + \sum_{i=-k+1}^{-1} \sigma_i^{2+} - (A_0^{2+} - A_{-k+1}^{2+}) \right) + (A_0^{2+} - A_0^2) \\ &\leq W_0^{2+} + (A_0^{2+} - A_0^2) \end{aligned} \quad (4.11)$$

due to the fact that W_{-k}^{2+} also satisfies Lindley's equation and $\forall x, x \leq \max(0, x)$ in Lindley's equation.

We can bound below \tilde{W}_j^{2-} in the same fashion by $(W_j^{2-} + (A^{2-} - A^{2+}))^+$. This completes the third step of our lemma's proof.

These last bounds are true almost surely, hence we can put everything together, and complete the proof of the lemma:

$$W_j^{2-} - (A_j^{2+} - A_j^{2-}) \leq_{\mathcal{D}} W_j^2 \leq_{\mathcal{D}} W_j^{2+} + (A_j^{2+} - A_j^{2-}) \quad (4.12)$$

In order to be able to iterate this construction to the next step, we need to define A_j^{3-} and A_j^{3+} ;

A_j^{3+} is the output process of the G/G/1 queue with arrival process A_j^{2+} and service times σ_j^{2+} (namely the SUBSystem); A_j^{3-} is the output process of the G/G/1 queue with arrival process A_j^{2-} and service times σ_j^2 (namely the SLBSystem).

It is easy to see that all processes converge to equilibrium, and that indeed $A_j^{3-} \leq A_j^3 \leq A_j^{3+}$: in the SUBS system, the customers arrive later and get served a longer time, hence depart even later than in the original system. Note that this is true of arrival processes, but not of the waiting time processes, since a delayed arrival might give the SUB system more time to empty.

■

The upper bound in the inequality 4.12 is not necessarily finite. We need to prove it is in order to prove Theorem 4.2.1. This is the object of the next lemma:

Lemma 4.2.2

$$A_j^{i+} - A_j^{i-} < \infty \text{ a.s.} \quad (4.13)$$

Proof:

We proceed by iteration: for $i = 1$, we can write:

$$A_j^{i+} = A_j^{i-} \text{ a.s.} \quad (4.14)$$

Now, knowing that $A_j^{i+} - A_j^{i-} < \infty$, we can write for the next step:

$$A_j^{i+1+} - A_j^{i+1-} = (A_j^{i+} - A_j^{i-})_{\mathfrak{S}\mathfrak{I}}^+ (W_j^{i+} - W_j^{i-}) + (\sigma_j^{i+} - \sigma_j^{i-}) \quad (4.15)$$

where σ_j^{i+} is defined as σ_j^i plus a geometric number of i.i.d. random variables finite almost surely. This is a sum of three almost finite random variables, thus this concludes the proof of this lemma.

■

Putting everything together, we are now able to state that the waiting time in our system is always bounded above by two terms:

- the waiting time of a G/G/1 system with arrival process converging to steady state and i.i.d. service sequence.
- a dispersion factor modeling the difference into the arrival process between the worst case and the best case situations.

Both these terms converge to steady state, hence we found a steady state distribution on the waiting time in the system.

4.3 I.I.D. General Network

4.3.1 Introduction

In this section, we extend naturally the previous result obtained in a particular and very simple network to the general topology described in Chapter 2.

A couple of general remarks are in order before we discuss the details more thoroughly. The proof we gave in the previous section took into consideration the fact that there was only one route in the system and that there were only two classes at each station.

We now consider the model we defined in Chapter 2, with the caveat that all the distributions are i.i.d. and mutually independent. The heuristics of the proof are that we can reproduce the induction from the particular Lu-Kumar case to each route of the general topology, and that the influence of more than one other class at each buffer will complicate the notations but not the reasoning.

4.3.2 General Case

We consider now a network with B buffers defined according to our model of chapter 2. We assume that the necessary stability condition for the network is satisfied, that is:

$$\forall s \in S, \sum_{b \in B: s_b = s} \lambda^b \bar{\sigma}^b < 1. \quad (4.16)$$

where a barred quantity denotes its expected value.

The scheduling policy works in the following fashion: every time station s becomes empty, it throws a coin pointing at buffer b such that $s_b = s$ with probability π_b^s . If buffer b is empty, it throws the coin again and again until it points to a non-empty buffer. If all buffers are empty, the next customer to arrive at station s directly goes into service. The coin tossing random variables are independent from the other stochastic processes, and identically distributed. We denote the outcome of the n th coin toss at station s by c_n^s .

We define the waiting time W_j^b to be the waiting time of the j th customer entering buffer b , namely the time spent between its arrival and its starting to receive service. We define the departure process from buffer b by d_j^b . d_j^b is the time at which the j th customer from buffer b departs station s . We also define a_j^b to be the arrival time of the j th customer into buffer b . Thus, for a buffer b_2 immediately following a buffer b_1 on a given route, $a_j^{b_2} = d_j^{b_1}$.

Now assume that we know that there exist two processes a_j^{b-} and a_j^{b+} such that $a_j^{b-} \leq a_j^b \leq a_j^{b+}$ where a_j^{b-} and a_j^{b+} converge to some stationary process with rate λ^b , and the difference $a_j^{b+} - a_j^{b-}$ is finite almost surely.

We define the upper bound system $b+$ to be a G/G/1 queue with arrival process a_j^{b+} and service time σ_j^{b+} . σ_j^{b+} is a service time depending on some more definitions. We denote by μ_k^β a sequence of random variables independent from the other stochastic processes such that μ_k^β is a geometric random variable with parameter $\frac{\pi_{s_b}^b}{\pi_{s_b}^\beta}$ defined for all buffers β such that $s_b = s_\beta$. Namely, μ_k^β corresponds to how many times the scheduling coin at station s_b points at station β between to occurrences of pointing at b . We denote by k_j^β , $-\infty < j < \infty$ a sequence of indices such that k_0^β is given, and $k_{j+1}^\beta = k_j^\beta + \mu_{k_j^\beta}^\beta$. We are ready to define

$$\sigma_j^{b+} = \sigma_j^b + \sum_{\beta \neq b: s_\beta = s_b} \sum_{k=k_j^\beta+1}^{k_{j+1}^\beta} \sigma_k^\beta. \quad (4.17)$$

σ_j^{b+} heuristically corresponds to the service time seen by a customer of class b in the original system, if the coin toss would never point at an empty buffer. It is i.i.d., due to the fact that all sequences involved in its definition are i.i.d., and we can compute its mean:

$$\bar{\sigma}^{b+} = \bar{\sigma}^b + \sum_{\beta \neq b: s_\beta = s_b} \frac{\pi_{s_b}^\beta}{\pi_{s_b}^b} \bar{\sigma}^\beta = \frac{1}{\lambda^b} \sum_{\beta: s_\beta = s_b} \lambda_\beta \bar{\sigma}^\beta \quad (4.18)$$

By the stability condition (4.16), (4.18), and the fact that a_j^{b+} has a rate λ^b , a G/G/1 queue with arrival process a_j^{b+} and service σ_j^{b+} is stable.

We define in a parallel manner the system $b-$ which is also a G/G/1 queue, with incoming customers arriving according to the process a_j^{b-} and getting service time $\sigma_j^{b-} = \sigma_j^b$. σ_j^{b-} corresponds to the case in the original system when the coin toss always point to an empty buffer but for customers in b . For the same reason as $b+$, $b-$ is stable.

For both systems $b+$ and $b-$ we also define the waiting times W_j^{b+} and W_j^{b-} .

Customers in systems $b+$ and $b-$ have different service times and arrival times than customers in the original system at buffer b . In order to make the comparison possible between say W_j^{b+} and W_j^b , we introduce two other systems, $\tilde{b}+$ and $\tilde{b}-$ respectively. $\tilde{b}+$ (resp. $\tilde{b}-$) is a G/G/1 queue with arrival process a_j^b and service time σ_j^{b+} (resp. σ_j^{b-}).

It should be obvious that with consistent notations:

$$\tilde{W}_j^{b-} \leq W_j^b \quad (4.19)$$

since the arrival times in both $\tilde{b}-$ and the original system coincide, but the service times are always less in $\tilde{b}-$.

Also,

$$\tilde{W}_j^{b-} \geq W_j^{b-} - (a_j^{b+} - a_j^{b-}) \quad (4.20)$$

this is a consequence of Lindley's equation according to the same steps as in the equations 4.11, hence, together with equation 4.19:

$$W_j^{b-} - (a_j^{b+} - a_j^{b-}) \leq W_j^b \quad (4.21)$$

In order to get a stability result, we want an upper bound. Now, we cannot write that $W_j^b \leq \tilde{W}_j^{b+}$ without discussing the relation between the μ_k^β 's, the k_j^β 's and the sequence of coin tosses c_n^s .

Since all sequences are i.i.d., we can construct (or couple) an outcome of the upper bound system with the exact same probability that satisfies the desired bound. We define it as in the particular case by first coupling the μ_k^β 's with the outcome of the coin toss sequence, then either copying the value of the actual service time if the coin toss points at a non-empty buffer, and an actual customer goes into service, or substituting an i.i.d. outcome of the same distribution when the coin points at an empty buffer.

Hence we attain a bound in distribution, since we construct an outcome of the Upper Bound system that is as general as can be, and which is a bound on the original system.

Consider now the system $b+$. It is another $G/G/1$ queue with i.i.d. arrival process and service times, hence converges to a steady-state distribution almost surely, which we denote W^{b+} .

Now, the relation 4.20 can be replicated with \tilde{W}_j^{b+} and W_j^{b+} , hence:

$$\tilde{W}_j^{b+} \leq W_j^{b+} + (a_j^{b+} - a_j^{b-}) \quad (4.22)$$

Now, this implies that $\tilde{W}^{b+} < \infty$ almost surely, that on the same sample path, $W^{b+} \leq W^{b+} + \delta$, where δ is a random variable with distribution the same as $a_j^{b+} - a_j^{b-}$.

Due to the coupling between the original system and the Upper bound system, we know that

$$W_j^b \leq_{\mathcal{D}} \tilde{W}_j^{b+} \quad (4.23)$$

and we can deduce from 4.22 that

$$W_j^b \leq_{\mathcal{D}} W^{b+} + \delta \quad (4.24)$$

To sum it up, we proved that under our assumptions on a^{b-} and a^{b+} ,

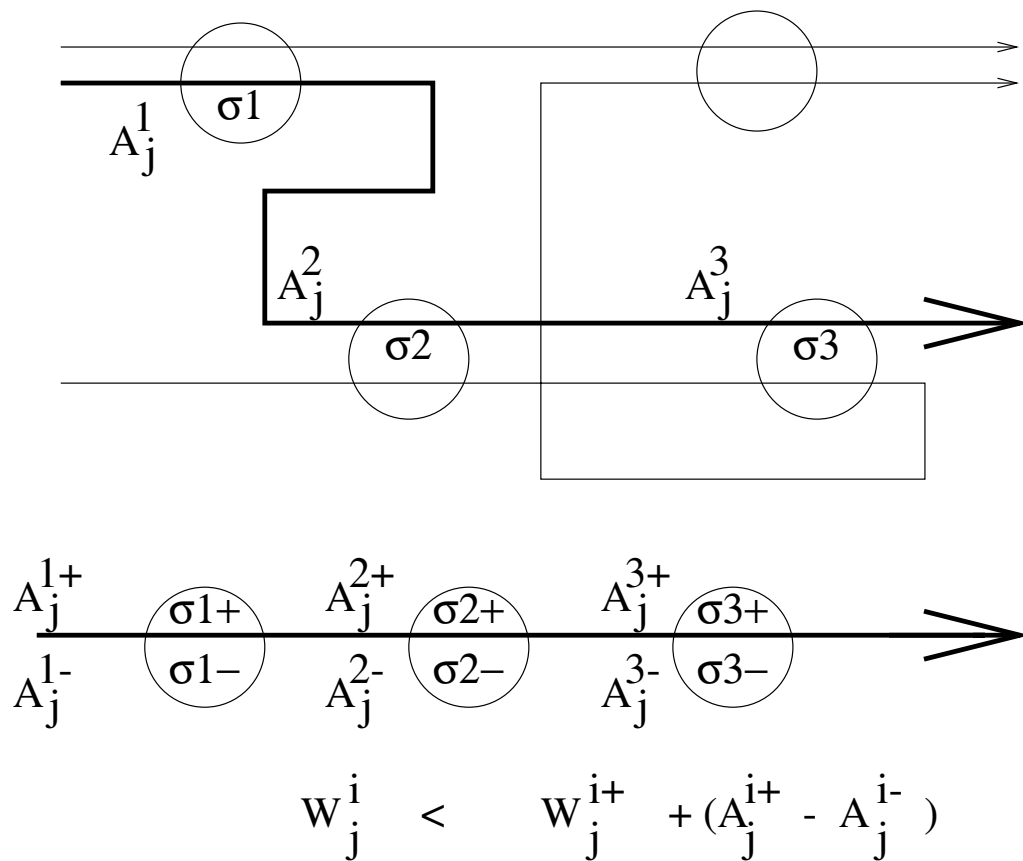
$$(W^{b-} - \delta_-)^+ \leq W_0^b \leq W^{b+} + \delta_+ \quad (4.25)$$

with δ_- and δ_+ with distribution δ .

Our assumptions are satisfied when considering external arrival processes in which case $a_j^{b+} = a_j^{b-} = a_j^b$ the arrival process of customers of class b into the network.

Now assuming that on a given route, all buffers b_1, b_2, \dots up to b_k (included) satisfy 4.25. Then the output process d^{b_k-} defined by the output of b_- and d^{b_k+} defined by the output of b_+ will satisfy the requirements. Hence b_{k+1} also satisfies 4.25

This completes the proof of our theorem.



Relation between a given route and a tandem queue

Figure 4.2: Extraction principle

4.3.3 Conclusion

We have proved that for a general network, the random multiplexing policy was strongly stable, and will study its Harris recurrence property in the Markovian case.

We have defined random variables which dominate the behavior of the state of the system. We also have underlined a decomposition principle, since the bounds correspond to the nodes in isolation fed with the output process of the previous node in isolation. Hence we are practically considering a tandem network. The random multiplexing policy asymptotically cancels the effect of feedback loops.

4.4 Harris recurrence

4.4.1 Introduction

In this part, we use the previous theorem to prove a stronger result: that the state of our i.i.d. network is Harris recurrent.

Harris recurrence is the property that there exists a measure such that if the set of states A has a positive measure, then the time for the system to hit A is finite with probability one. Also, positive Harris recurrence implies that the process has a unique invariant measure to which the process converges. We refer the reader to Dai [19] for a more thorough discussion of Harris recurrence.

Our purpose here is not to dwell on Harris recurrence, but to use some results proved by Dai: namely, we will check our network satisfies the hypotheses of one of Dai's theorem, and conclude that our process is Harris recurrent. Basically, it means that the state of the system converges to a stationary distribution, and that the stability of the system is the strongest we can hope for.

Also, we stated in the previous section that we could bound the waiting time and the behavior of our system by two terms: one being some $G/G/1$ queues in tandems, the other being some dispersion parameter that gives an indication of how much the original system can stray compared to the $G/G/1$ queues in tandem.

G/G/1 queues in tandem are very easy to study: namely such a network converges to a steady state. If we look at the fluid model of such a network, it is straightforward, since there is no feedback loop, and there is no conflict for service resource. The fluid model of such a network is stable.

Our goal is to deduce from the fluid model stability of the G/G/1 queues in tandem that the fluid limit model of our Chapter 2 i.i.d. network is also stable in any set of initial conditions, thus proving by making use of one of Dai's theorem that our network is positive Harris recurrent and converging to its own steady state.

4.4.2 Proof of the Harris recurrence

First, let's recall the result we want to use, which can be found in Dai, [19]:

Theorem 4.4.1 *If the system satisfies*

- (i) *the arrival and service sequences are i.i.d. and mutually independent*
- (ii) *the necessary stability condition $\rho < 1$ is satisfied at each station*
- (iii) $P(\tau_0^i \geq x) > 0, \forall x > 0$ (4.26)

and the fluid limit model of the queueing discipline is stable, then the Markov chain describing the dynamics of the network under the discipline is positive Harris recurrent.

We can impose the conditions (i),(ii),(iii) on our network. (ii) is necessary to have any kind of stability, since it says the average quantity of work per unit of time incoming at any station has to be less than one. (i) is the general assumption we make in this chapter. (iii) is a new condition we make, that the arrivals are spread out in time.

We need now to explain a little bit what is the fluid limit model, and to prove whether or not the fluid limit model is stable.

4.4.3 Fluid Model

Fluid models are useful tools for studying the stability of networks. Indeed, by rescaling the time and the load in the network, they only consider the deterministic behavior of a limit instead of the random behavior of a sample path.

Recall we have B buffers numbered from $1, \dots, B$, and S stations. The state of the network is described by a Markov process $X = \{X(t), t \geq 0\}$, where $X(t)$ is the vector:

$$X(t) = (Q_1(t), \dots, Q_B(t), C_1(t), \dots, C_S(t), U_1(t), \dots, U_S(t), V_1(t), \dots, V_B(t)) \quad (4.27)$$

where $Q_b(t)$ is the number of customers in class (or buffer) b at time t , $C_s(t)$ is the class of the buffer being served at station s at time t , $U_s(t)$ is the remaining service time at station s at time t , and $V_b(t)$ is the remaining interarrival time until the next external arrival in buffer b .

Under the Random Multiplexing policy, this is a Markov process and suffices to describe the state of the system, under the assumption we make that all the processes are i.i.d. and mutually independent. We refer the reader to Durrett for a discussion of Markov processes [27]. It is easy to see that due to the iid property, the state of the system at any future time depends only on the present state, but not on the past history of the system.

The fluid limit of a quantity $Q(t)$ is the limit of the sequence:

$$\frac{1}{n}Q^n(nt) = \bar{Q}(t) \quad (4.28)$$

where $Q^n(\cdot)$ is the quantity starting from the n^{th} initial condition x_n , when the initial condition grows such that $\frac{|x_n|}{n} \rightarrow 1$.

The fluid limit is stable if for all solution of the fluid limit model, $\bar{Q}(t) = 0$ for t greater than some $\delta > 0$.

We can state our main result:

Theorem 4.4.2 *The fluid limit model of the Chapter 2 model in the i.i.d. case is stable*

Proof: Now, in our situation, the number of customers at buffer b is given by:

$$Q_b(t) = Q_b(0) + \sum_{i=0}^{\infty} \mathbf{1}_{\{a_i^b < t\}} - \sum_{i=0}^{\infty} \mathbf{1}_{\{d_i^b < t\}} \quad (4.29)$$

We can use the quantities $a_i^{b\pm}$ and $d_i^{b\pm}$:

$$Q_b(t) \leq \left(Q_b(0) + \sum_{i=0}^{\infty} \mathbf{1}_{\{a_i^{b-} < t\}} - \sum_{i=0}^{\infty} \mathbf{1}_{\{d_i^{b-} < t\}} \right) + \left(\sum_{i=0}^{\infty} \mathbf{1}_{\{d_i^{b-} < t\}} - \sum_{i=0}^{\infty} \mathbf{1}_{\{d_i^{b+} < t\}} \right) \quad (4.30)$$

The upper bound is composed of two terms in between the parentheses: the first term corresponds to the number of customers in the system we denoted by $b-$, namely the stationary lower bound system: it is a G/G/1 queue, hence, whenever we do the fluid scaling, we know there exist some value δ such that for any $t > \delta$, the fluid scaling of this quantity is equal to 0, and this independently of the initial condition.

The second term in the upper bound converges to zero for any fluid scaling: to better understand this claim, remember that d^{b+} and d^{b-} are two processes which converge to a steady state, and whose difference is finite almost surely.

Taking the last two paragraphs into account, we can thus write that there exists δ_b such that:

$$\lim \frac{Q_b^n(nt)}{n} = 0, \quad \forall t > \delta_b > 0 \quad (4.31)$$

This is true for all buffers, hence taking the maximum over all δ_b 's: $\delta = \max_{b=1, \dots, B} \delta_b$, yields the stability of the fluid limit model. This is turn is enough to ensure that the network is positive Harris recurrent by Theorem 4.4.1.

Harris recurrence within this setup then implies again the convergence to the steady state of the Markov chain.

4.5 Conclusion

We have proved in this chapter that the stability domain of any Markovian network topology was the domain $\rho < 1$, even though we imposed the stability to be the convergence to an almost surely finite steady state.

This result is of tremendous importance: it states that for any network, in a Markovian framework, the random multiplexing scheduling policy will yield the best throughput and the convergence to a steady state. It is then easy to dimension such a network for instance, by evaluating the number of customers in each buffer.

The result is that of a decomposition principle, in which each route is analyzed separately. The analysis of such a tandem network is well known and not too difficult. The waiting

time of a customer can itself be decomposed into two terms, that of a queue in a tandem, and that of some dispersion factor.

We also proved that for any network topology, the state of the system would converge to equilibrium, independently of the initial conditions. The random multiplexing policy thus is optimal, and not only sub-optimal as in the previous chapter.

Chapter 5

Robustness

5.1 Introduction

We have seen so far that the stability domain of any network topology satisfying our model was maximized to $\rho < 1$ for the random multiplexing scheduling policy. We have proved that this was true in the rate stable sense as well as in a strongly stable sense.

Now the question we would like to investigate is that of the robustness of such a policy. It is very nice to have a theoretic result as to what is the stability domain of a network. Yet it is quite useless if a tiny variation in one of the parameters throws the network off.

There are two ways of answering the robustness question.

- One is to define a stability domain for the scheduling policy. That is, for a given set of parameters, we can construct a stable scheduling policy. If we freeze this scheduling policy, the question is what are now the admissible parameters that will still keep the overall property of stability for the network. If we can prove that for a given set of parameters, the policy we fix has a stability region that admits perturbations of these parameters that were used to define it, then we would have a robustness result. This will be the object of the next section.
- The other way of answering the robustness issue is to define another scheduling policy that adapts itself to perturbations in the network. Even more robust would be a policy that learns from the network behavior in order to define itself. Assuming for instance

that the number of buffers at each station is fixed, we can admit perturbations in the routing parameters, the flow rates, or the service rates. We will turn our attention to such a policy in the last two sections of this chapter.

We will first define such an adaptive scheduling policy, and prove it is rate-stable. Then in the following section, we will also prove it is strongly stable in a Markovian framework.

5.2 Admissible parameters for a fixed random multiplexing policy

5.2.1 Introduction

Since the stability is an asymptotic result, we have to make the assumption that from some point on in time, the model assumptions are satisfied.

Recall the way our random multiplexing scheduling policy was defined: to make a scheduling decision at station s , a coin was tossed pointing at buffer $b \in s$ with probability π_b^s such that

$$\pi_b^s = \frac{\lambda_b}{\sum_{\beta \in s} \lambda_\beta} \quad (5.1)$$

As we can see, the probability distribution of the coin toss depends only on the rates of customers arriving at the buffers of the station.

This answers a first question: as long as the arrival rates do not suffer any perturbation, the random multiplexing policy is stable provided that the stability condition $\rho < 1$ is still satisfied.

Also, as long as the ratios of the arrival rates do not vary, namely as long as λ_b/λ_β is constant for all couples (b, β) , and the stability condition $\rho < 1$ is still satisfied, then the random multiplexing policy is stable.

Remark: re-entrant lines. Based on the previous paragraph, we can state that in a Re-Entrant lines, the random multiplexing policy is not sensitive to arrival rates perturbation, provided $\rho < 1$ at each station. Recall that a re-entrant line is a multiclass network in which customers follow one single route, thus the ratios of arrival rates to each station are constant.

This also raises the next question: would a perturbation on the arrival rates lead the system to instability? This question is of practical interest in many applications: the service requests do not vary much, since they depend on the ability of the station to provide such a service. The statistics of the size of the packets coming to a router are going to be more or less the same. The distribution of the time a silicon wafer has to be baked in an oven does not vary much. It is the arrival rates that may be subject to variation, since we may want to increase the throughput in a network, for instance, or a link going down somewhere might add some load on some other link across the network.

We prove in this section a strong stability result, namely that the state of the system converges to a steady-state distribution in a Markovian framework. Also, in order not to be too repetitive, we do not give the proof of the rate stability in a stationary and ergodic framework. Yet, the astute reader would be able to apply the same mapping from the proof of the strong stability in Chapter 3 and this result, as to the proof of the rate stability in Chapter 2 and the equivalent rate stability robustness result.

We give ourselves a vector $\vec{\Lambda} = (\Lambda^1, \Lambda^2, \dots, \Lambda^r)$ of arrival rates of customers to the r routes of the network, and define the corresponding random multiplexing scheduling policy $\Phi(\vec{\Lambda})$. We refer ourselves again to the model of chapter 2, and relax the i.i.d. assumption for the sequences that we made in chapter 4.

We will prove that there is an open ball around $\vec{\Lambda}$ such that the random multiplexing policy is stable. We give a bound on the variation around $\vec{\Lambda}$ that ensures stability of $\Phi(\vec{\Lambda})$.

Theorem 5.2.1 *There is a ball $B(\vec{\Lambda}, \epsilon)$ of positive radius ϵ such that:*

$$\forall \lambda \in B(\vec{\Lambda}, \epsilon), \Phi(\vec{\Lambda}) \text{ is stable} \tag{5.2}$$

Proof:

We consider here again the Markovian framework of the previous chapter, that is the identically distributed and independent driving sequences of the system. The proof is done for station s . It is another example of the independence principle of the random multiplexing policy. It builds on the proof of the strong stability of the random multiplexing policy described in the previous chapter. The outline of the proof is as follows: we assume that the arrival process at a given buffer b follows some nice property, with the arrival rate at buffer b being λ_b instead of Λ_b . We prove that the departure process of customers of class b from station s is such that:

- (i) the waiting time of a customer of class b is bounded above in distribution, by a distribution that ensures the strong stability,
- (ii) the departure process of customer of class b from station s also satisfies the same nice property.

From then on, it is easy to see that we can extend the result to all buffers and all stations, hence having the desired result. Indeed, if we can define at each station a neighborhood $B(\vec{\Lambda}, \epsilon_s)$ such that $\Phi(\vec{\Lambda})$ is stable and $\epsilon_s > 0$, then by taking $B(\vec{\Lambda}, \epsilon) = \bigcap_{\text{all } s} B(\vec{\Lambda}, \epsilon_s)$ we complete the proof of theorem 5.2

The intuition tells us that if we decrease the value of the rates of arrival of customers into the system and keep the same service requirements, we decrease the load on the system, and we expect to conserve the stability of the system. We know ([24]) that this is not true for a general policy: it is another positive aspect of the random multiplexing to ensure this rather intuitive property. On the other hand, we will also prove that we can increase the rates of arrival. There exists, for a given scheduling policy and service requirements, a set of admissible arrival rates.

We will answer at the same time a reverse problem: since if the arrival rates can vary for a fixed policy, it means that the same policy is stable for different arrival rates. The reverse problem is to look at which scheduling policies, of the form described in our model, are stabilizing our model under a fixed set of arrival rates and service requirements.

5.2.2 Example

We start this discussion by a very simple example. Consider again the Lu-Kumar network: it is a 2 buffer 4 station fixed route network. We assume the arrival rate to be λ for the arrival sequence with interarrival time τ_j and the service requirement at buffer 1,2,3,4, which customers visit in this order, to be $\sigma_j^1, \sigma_j^2, \sigma_j^3, \sigma_j^4$. With the same notations as previously we have the necessary stability condition:

$$\lambda(\bar{\sigma}^1 + \bar{\sigma}^4) < 1; \lambda(\bar{\sigma}^2 + \bar{\sigma}^3) < 1 \tag{5.3}$$

Recall that in this framework, we have proved that the random multiplexing policy with probability distribution of picking buffer 1 or 4 at station 1 picked to be $\frac{\lambda}{\lambda+\lambda} = \frac{1}{2}$ for both buffers, and with probability distribution one half, one half of picking buffer 2 or 3 at station 2, is stable.

Consider a scheduling policy following the random multiplexing principle described in the chapter 2 with the probability distributions given now by:

$$\begin{aligned}\pi_4^1 + \pi_1^1 &= 1; \pi_3^2 + \pi_2^2 = 1; \\ 0 &\leq \pi_1^1, \pi_4^1, \pi_2^2, \pi_3^2 \leq 1\end{aligned}\tag{5.4}$$

Customers arriving at buffer 1 are delayed by customers already in buffer 1 and customers being picked from buffer 4. If we can bound the waiting time of a customer from buffer 1, then buffer 1 would be stable. The waiting time of a customer j from buffer 1 is less than the waiting time of a customer from a G/G/1 system in which customers are served a service time:

$$\Sigma_j = \sigma_j^1 + \sum_{k=\alpha_j}^{\gamma_j} \sigma_k^4\tag{5.5}$$

where $\alpha_{j+1} = \gamma_j + 1$ and $\gamma_j - \alpha_j$ is a geometric random variable corresponding to the number of times the scheduling coin toss at station 1 points at buffer 4 before pointing at buffer 1 after completing the service of customer $j - 1$ from buffer 1.

We pick α_0 such that the set of indice $\gamma_j - \alpha_0$ includes all the customers from buffer 4 actually served in between customers 0 and customer j of buffer 1.

This correspond to the worst case situation for customer of buffer 1, namely every time a customer from buffer 4 is called ahead of customer of buffer 1, it is present and receives its service. It is also an impossible situation in many situations. If for instance π_4^1 is higher than π_1^1 , there are more customers from buffer 4 called to receive service than customers from buffer 1: this is impossible since each customer from buffer 4 in this network has to be first a customer from buffer 1.

Σ_j is a independent and identically distributed sequence, since σ_j^1 and σ_j^4 are both i.i.d. sequences and the coin toss is independent from those two sequences and i.i.d. The mean of Σ_j is, due to the distribution of the coin tosses:

$$\bar{\Sigma} = \bar{\sigma}^1 + \frac{\pi_4^1}{\pi_1^1} \bar{\sigma}^4\tag{5.6}$$

No matter how α_0 is chosen, a G/G/1 system with the service structure Σ_j and the arrival process τ_j is stable and converges to a stationary regime (by Loynes) if and only if:

$$\lambda \bar{\Sigma} < 1\tag{5.7}$$

This in turn is true in our example if:

$$\lambda\bar{\sigma}^1 + \frac{\pi_4^1}{\pi_1^1}\lambda\bar{\sigma}^4 < 1 \quad (5.8)$$

One can see that taking $\pi_4^1 = \pi_1^1 = \frac{1}{2}$ as in the previous chapters, would give back the stability condition, which is necessarily true. Since $\pi_4^1 = 1 - \pi_1^1$, this yields a relation on π_1^1 :

$$\pi_1^1 > \frac{\lambda\bar{\sigma}^4}{1 + \lambda(\bar{\sigma}^4 - \bar{\sigma}^1)} \quad (5.9)$$

An upper bound on π_1^1 can be found by considering the worst case situation for buffer 4. By considering the worst case situation for buffer 2 and 3, we can also find an interval of values that ensures stability for π_2^2 (and consequently π_4^1 and π_3^3).

If on the contrary, we assume that $\pi_1^1 = \pi_4^1 = \pi_2^2 = \pi_3^3 = 1/2$ then we are of course in the stability interval. The admissible values of λ are those that satisfy equation 5.8. Since we were considering a worst case example, equation 5.8 is a sufficient condition. It turns out to be a necessary condition in this particular example due to the fact there is only one dimension in the vector of arrival rates.

5.2.3 General case

In this section, we extend the result that was illustrated in the previous example to our general model. We consider the model we defined in Chapter 2 with the difference that the π_b^s 's are now fixed arbitrarily, and not as a function of the arrival rates λ_b anymore, and with the added constraint of all sequences being i.i.d. Still for all s , and for all $b \in s$, the π_b^s must define a probability distribution.

We consider the input to the network as constructed in the sequence $IN_j, -\infty < j < \infty$. We want to prove the following result:

Theorem 5.2.2 *A sufficient condition for stability of the network is that:*

$$\forall s \in S, \forall b \in s, \lambda_b \left(\sum_{\beta \in s} \frac{\pi_\beta^s}{\pi_b^s} \bar{\sigma}^\beta \right) < 1 \quad (5.10)$$

Note that taking the λ_b 's such that for all s and b :

$$\pi_b^s = \frac{\lambda_b}{\sum_{\beta \in s} \lambda_\beta} \quad (5.11)$$

turns the sufficient condition 5.10 into the necessary stability condition $\rho < 1$. Hence this chapter includes the previous chapter which included already its previous chapter.

Remark: This proves also that there always exists a solution to this set of inequalities; hence, for a given set of arrival rates $\vec{\Lambda}$, there is an open set in the Π space (the space of vectors π_b^s) such that the system is stable.

Now we assume that we can bound the arrival time a_j^b of customer j at buffer b by the arrival processes a_j^{b-} and a_j^{b+} such that

$$\begin{aligned} (i) \quad & a_j^{b-} \leq a_j^b \leq a_j^{b+} \\ (ii) \quad & a_j^{b+} \text{ and } a_j^{b-} \text{ converge to some steady state distribution as } j \rightarrow \infty \\ (iii) \quad & a_j^{b+} - a_j^{b-} < \infty \text{ a.s.} \end{aligned} \tag{5.12}$$

The conditions (i), (ii), (iii) are referred to as the arrival conditions 5.12.

It is obvious that the first buffer in any route is going to satisfy these conditions for $a_j^{b-} = a_j^b = a_j^{b+}$.

Consider now an arbitrary buffer b of station s and let the customers flow in the system from time $-\infty$. Consider the busy period of buffer b (as defined previously) leading to customer 0. The waiting time of customer 0 is the waiting time accumulated during this busy period of buffer b . It corresponds of the customers from buffer b that are already in the system at the arrival time of customer 0, and that of the customers from the buffers $\beta \in s$ that are picked by the scheduling coin tosses to be served ahead of customer $(0, b)$.

The worst case situation for customer $(0, b)$ is that all customers from other buffers are present when called upon. Hence the waiting time of customer 0 is less than the waiting time of customer 0 from the G/G/1 system with arrival process a_j^b and service requests:

$$\Sigma_j^b = \sigma_j^b + \sum_{\beta \in s} \sum_{k=\alpha_j^\beta}^{\gamma_j^\beta} \sigma_k^\beta \tag{5.13}$$

where $\alpha_{j+1}^\beta = \gamma_j^\beta + 1$ and $\gamma_j^\beta - \alpha_j^\beta$ correspond to the number of coin tosses by the scheduling coin pointing at buffer β before pointing back at customer b after servicing customer $(j-1, b)$. Hence it $\gamma_j^\beta - \alpha_j^\beta$ is a geometric random variable with mean $\frac{\pi_\beta^s}{\pi_b^s}$. Also, we pick α_0^β such that all customers from buffer β actually served during the busy period of buffer b leading to customer 0 have their indices in the interval $[\alpha_{j_o}^\beta, \gamma_0^\beta]$ where j_o is the index of the first customer of the busy period of buffer b leading to customer 0 ($j_o \in (-\infty, 0)$).

As discussed earlier in chapter 4, the stability of the model is conditioned by the stability of the G/G/1 system (a_j^{b+}, Σ_j^b) , which in turn is satisfied iff:

$$\lambda_b \bar{\Sigma}^b < 1 \tag{5.14}$$

By the assumption 5.10, this is true.

The same discussion as in chapter 4 allows us to construct the departure processes a_j^{b+1-} and a_j^{b+1+} satisfying the conditions 5.12, where $b + 1$ is the buffer following b on the route.

■

Now, we illustrate in figure 5.1 the area of admissible parameters for a 2 classes network.

This answers the two questions we mentioned earlier: since the domain given by the sufficient condition is an open domain, we can always find a neighborhood of a point $\vec{\Lambda}$ that is stable.

Also, for a point $\vec{\Lambda}$, we can compute the set of admissible scheduling policies that are stable, in the form of open intervals around the value $\lambda_b^s / (\sum_{\beta \in s} \lambda_\beta^s)$ for π_b^s .

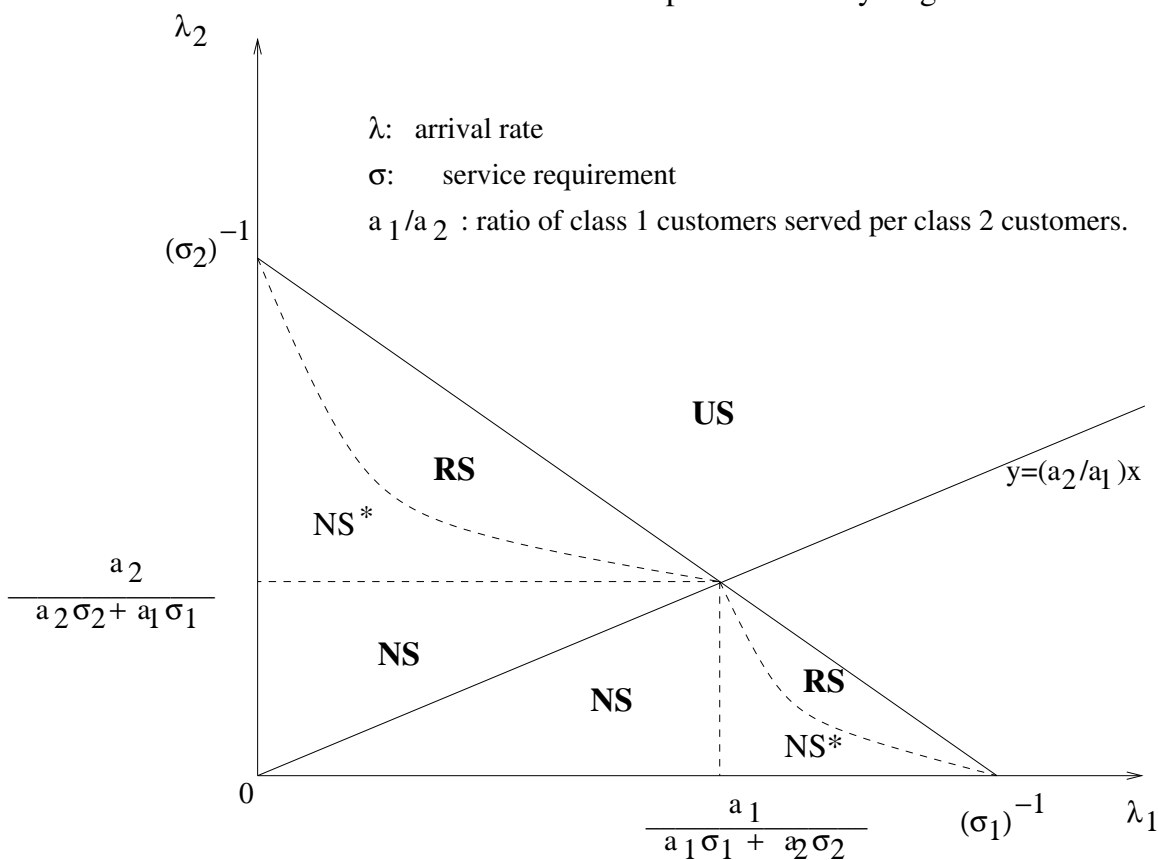
This concludes our discussion of the robustness of the scheduling policy defined in the chapter 2. We now turn our attention to a new scheduling policy, for which we aim at proving the same results as before, namely rate-stability, strong stability and robustness.

5.3 Rate stability of the adaptive multiplexing policy

The scheduling policy we have discussed so far requires the station to know the average number of customers of a given class in order to compute the distribution of the scheduling coin toss. We now define a new scheduling policy, based on the Statistical Multiplexing policy, which does not require this a priori knowledge. We will prove this policy is also rate-stable.

Note that in a system where the conditions change over time, while always respecting the stability criterion 4.16, the Statistical Multiplexing Policy might not be optimal (or stable) anymore. This leads us to consider an Adaptive Statistical Multiplexing.

A 2-class example for Stability Regions



US: unstable area; RS: can only guarantee router stability
 NS: guarantees Network Stability as well. (* conjectured)

Figure 5.1: Stability domain of a fixed random multiplexing policy

We define the Adaptive Statistical Multiplexing scheduling policy to behave exactly in the same manner as the Statistical Multiplexing policy but for one difference: the distribution of the coin tosses are not i.i.d. anymore, but independent, with a probability distribution $\pi_s^b(t)$ for a coin tossed pointing to buffer b of station s at time t , depending on time in the following manner:

$$\pi_s^b(t) = \frac{1}{\sum_{\beta: s_\beta = s_b} 1}$$

prior to the arrival of a customer to buffer b

$$\pi_s^b(t) = \frac{\sum_{j=0}^{\infty} \mathbf{1}_{\{a_j^b < t\}}}{\sum_{\{\beta: s_\beta = s_b\}} \sum_{j=0}^{\infty} \mathbf{1}_{\{a_j^\beta < t\}}}$$

after the first arrival to buffer b

(5.15)

Note that $\pi_s^b(t)$ is a random variable on the sigma algebra \mathcal{F}_t generated by the input and the coin tosses up to time t . Every time a station needs to make a decision, it computes the values of the $\pi_s^b(t)$'s, and throw the coin accordingly. We assume that this operation does not consume any time. t is a parameter in the $\pi_s^b(t)$'s, but is expressed only in order to underline the variation of the distribution with time. The station does not need to know the time, but just needs to have an account of the number of arrivals up until the decision epochs.

Some properties of the $\pi_s^b(t)$ need to be explained. Assume that route $r \in \mathcal{R}$ is rate stable, then:

Proposition 5.3.1 *There exists some $T > 0$ such that:*

$$\forall b \in \mathcal{R}, t > T, \beta \text{ such that } s_\beta = s_b$$

$$(i) \quad \pi_{s_b}^b(t) \geq \pi_{s_b}^b$$

$$(ii) \quad \frac{\pi_{s_b}^\beta(t)}{\pi_{s_b}^b(t)} \leq \frac{\pi_{s_b}^\beta}{\pi_{s_b}^b}$$
(5.16)

Proof: Those two inequalities stem from the fact that r is stable, and the other routes are not necessarily stable. Hence, the number of customers reaching s_b from another route, say r' , is less than the number of arrival to that route in the system. $\pi_{s_b}^b$ is the limit ratio of arrivals to buffer b over all *possible* arrivals to station s_b . Due to possible shortage of customers from route r' , the ratio $\pi_{s_b}^b(t)$ has to be at least equal to the value $\pi_{s_b}^b$. The second inequality holds through the same considerations. ■

In order for the $\pi_{s_b}^b(t)$ to behave in a smooth fashion, we add the hypothesis on the input stream that:

$$\sigma_j^b > \epsilon, \text{ and } \tau_j^r > \epsilon, \tag{5.17}$$

for some $\epsilon > 0$. This is not really a restriction, since we can consider ϵ as small as we want, and there is always some kind of handling or processing time for any task at any station.

Note that the systems described in the introduction would still exhibit the same stability behavior when replacing the 0 unit of times services by any ϵ neglectible with respect to the other service times.

Proposition 5.3.2 *Under hypothesis 5.17, and given $t_1 < t_2$ such that $\frac{t_2}{t_1} \rightarrow 1$, we have:*

$$\lim_{t_1 \rightarrow \infty} (\pi_{s_b}^b(t_2) - \pi_{s_b}^b(t_1)) = 0 \quad (5.18)$$

Proof: This comes from the fact that the arrival of customers to any buffers will vary at most linearly over a finite interval of time, due to 5.17. Hence, since $t_2 - t_1 = o(t_1)$, and due to the structure of the function $\pi_s^b(\cdot)$, for t_1 big enough, the variation in $\pi_{s_b}^b(\cdot)$ is as small as we want. ■

We are now in position to state:

Theorem 5.3.1 *The Adaptive Statistical Multiplexing defined above is rate-stable.*

Proof: Using the same notations and methods as in Chapter 3, it is easy to see that we are only interested in proving by induction on the stations along a given route of the network that the corresponding tilde system is stable.

The tilde system is again a tandem network. The corresponding tilde station for the first station of the route $r = \{1, 2, \dots, v\} \in \mathcal{R}$ is as follows:

The arrival stream is the same as in the original system arriving into buffer 1. The service policy is coupled with the service policy of the original system in the following fashion: for each customer index j we denote by f_j and l_j the first index and the last index of the busy period of buffer 1 of the original system containing j . Hence $f_j \leq j \leq l_j$. Recall that we call a buffer 1 busy period of the original system the sequence of consecutive indices containing j of customers from buffer 1 so that whenever the scheduling coin toss points at buffer 1, a customer is present in buffer 1.

The tilde station 1, upon arrival of customer f_j , couples its scheduling coin tosses to those of the original system upon arrival of customer f_j . Hence the tilde system replicates the coin tosses happening in the original system during the busy period $[f_j, l_j]$. The station then serves a customer if the coin toss is pointing at buffer 1 and a customer is waiting, or idles $\bar{\sigma}^1$ units of time if pointing at buffer 1 and no customer is waiting, or idles $\bar{\sigma}^b$ units of time if the coin is pointing at buffer $b \neq 1$.

After the sequence of coin tosses from the busy period $[f_j, l_j]$ is used (that is in the case where customers from buffer 1 have been called but where not present in the tilde station

or in between busy periods), we just use an i.i.d. coin distributed according to $\pi_i^{s_1}$'s defined as in Section 2.

The tilde station is going to be stable if and only if the service request of a customer plus the subsequent idleness imposed by the coin pointing at other buffers than buffer 1 is less than the interarrival time. Proving this last point will conclude the proof of Theorem 5.3.1 since we have seen that the stability of the tilde system is equivalent to the rate stability of the original system, by bounding the departure times of the original system as previously, and inducting on the steps of the route.

Now, we can write the total service request of customer j as:

$$\sigma_j^1 + \sum_{\{b \neq 1: s_b = s_1\}} \mu_j^b \bar{\sigma}^b \quad (5.19)$$

where μ_j^b is a random variable describing the number of times after completing service of customer j that the coin toss points at buffer b before pointing back at buffer 1. All we need to prove is that $E[\mu_j^b] \leq \frac{\pi^b}{\pi^1} = \frac{\lambda_b}{\lambda_1}$, so that the averaged expression in 5.19 be less than λ_1^{-1} by 4.16.

Define the sequence $c_n(j), n \in \mathbf{N}$ to be the coin tosses subsequent to the service completion of customer j until the coin toss points at buffer 1 again. $c_n(j)$ is a sequence of independent random variables, but their distribution varies. For simplicity sake, we will denote by $\pi_n^b(j)$ the distribution of $c_n(j)$ over the buffers $b \in \{b: s_b = s_1\}$ and π^b the static distribution $\pi_{s_b}^b$. We can now express μ_j^b on the set $SN = \{\sum_{\beta: s_\beta = s_1} \mu_j^\beta \leq N\}$, for some $N > 0$.

$$\begin{aligned} (\mu_j^b | SN) &= \sum_{n=0}^N \mathbf{1}_{\{c_n(j)=b; c_0(j), c_1(j), \dots, c_{n-1}(j) \neq 1\}} \\ \text{or:} & \\ E[\mu_j^b | SN] &= \sum_{n=0}^N P[c_n(j) = b; c_0(j), \dots, c_{n-1}(j) \neq 1] \\ (1) &= \sum_{n=0}^N P[c_n(j) = b] \prod_{i=0}^{n-1} P[c_i(j) \neq 1] \\ (2) &= \sum_{n=0}^N \pi_n^b(j) \prod_{i=0}^{n-1} (1 - \pi_i^1(j)) \\ (3) &\sim \sum_{n=0}^N \pi_0^b(j) (1 - \pi_0^1(j))^n \end{aligned}$$

$$\begin{aligned}
(4) &\leq \sum_{n=0}^{\infty} \pi_0^b(j) (1 - \pi_0^1(j))^n \\
(5) &= \frac{\pi_0^b(j)}{\pi_0^1(j)} \\
(6) &\leq \frac{\pi_{s_b}^b}{\pi_{s_1}^1} \tag{5.20}
\end{aligned}$$

The step (1) uses the independence of the coin tosses, step (3) uses proposition 5.3.2, where \sim means that we can achieve a difference as small as we want by considering t big enough (t is the time of the first decision epoch, the time of the N th decision epoch is $o(t)$ by Birckoff's ergodic theorem applied to the service times). Step (6) flows from 5.3.1.

Since $\sum_{\beta: s_\beta = s_1} \mu_j^\beta$ is finite with probability one, we can extend the last result to $E[\mu_j^b]$ by taking $N \rightarrow \infty$. This proves the stability of the tilde system, and completes the proof of Theorem 5.3.1. ■

5.3.1 Markovian case

We consider now the Markovian case, when all the sequences are i.i.d. We need to refine our description of the system by including in the state of the system the number of arrivals to each buffer, $N_b(t)$ number of total arrivals to buffer b . Including this, the N -uplet:

$$X(t) = (Q_1(t), \dots, Q_B(t), C_1(t), \dots, C_S(t), U_1, \dots, U_S, V_1, \dots, V_B, N_1(t), \dots, N_B(t)) \tag{5.21}$$

is a Markov process.

Using the same coupling as in chapter 4, we can prove an upper bound on the waiting time of a customer in the system, based on some relationship to a tandem system. This tandem system also has a fluid limit that is stable, hence we can duplicate the fluid model analysis to obtain:

Theorem 5.3.2 *The Dynamic Random Multiplexing converges to a steady state in a Markovian network.*

Proof:

There are two steps in this proof: We need to define a system that will yield the upper bound, and to prove that this system indeed has a fluid limit model that is stable.

Upper bound model: The dynamically adaptive random multiplexing policy has the property that at station s , the coin toss is distributed according to the distribution $\pi_s^\beta(t)$ at some time epoch t . Now we would like to find a bound on the waiting time of customers in buffer b , knowing that the arrival process to buffer b is stable.

We have seen in Proposition 5.3.1, we have:

$$\begin{aligned} \pi_s^b(t) &\geq \pi_s^b \\ \frac{\pi_s^\beta(t)}{\pi_s^b(t)} &\leq \frac{\pi_s^\beta}{\pi_s^b} \end{aligned} \quad (5.22)$$

for a time t long enough. This means that the number of times that the coin toss points at β in s between two times pointing at buffer b is less in distribution than the number of times a coin toss distributed according to the π_s^b instead of the $\pi_s^\beta(t)$ would point at the same buffer β . Denote by $\mu_j^\beta(t)$ the number of times the coin toss points at buffer β in between pointing at customer j and $j + 1$ of buffer b . We thus can construct μ_j^β that is a geometric random variable with mean $\frac{\pi_s^\beta}{\pi_s^b}$ such that for a fixed ω , $\mu_j^\beta(t) \leq \mu_j^\beta$.

By construction, the μ_j^β are extracted from an i.i.d. sequence. We can then construct a service time:

$$\sigma_j^{b+} = \sigma_j^b + \sum_{\beta: s_\beta = s} \sum_{i=1}^{\mu_j^\beta} \chi_i^\beta \quad (5.23)$$

where χ_i^β are extracted from the same distribution as σ_0^β with the coupling condition that for every customer k in buffer β between j and $j + 1$ of buffer b that is actually served when called upon, then $\chi_i^\beta = \sigma_k^\beta$. This coupling is possible since both r.v.'s have the same distribution and are i.i.d.

Now we see that we can use σ^{b+} as in the previous chapter, since it is an i.i.d. sequence of service times that bound the actual sequence of service times above. All the next steps follow exactly as in the proof of the upper bounds in the Markovian framework.

Stability of the fluid limit model: The stability of the fluid limit model is due to the fact that the upper bound model correspond to a some iid queues in tandem, hence it is stable by the same argument as in the static case.

We can now conclude the proof since all the other steps are exactly the same as in Chapter 4.



5.3.2 Conclusion

We said we would prove the same results for the dynamic random multiplexing as for the random multiplexing scheduling policy, namely rate stability, strong stability and robustness.

We did prove in this chapter the rate-stability, and the convergence to steady state in a Markovian framework. And since the dynamic random multiplexing policy adapts itself to the changes in the network, we do not have to worry about robustness.

This result is important since it implies that by putting together black boxes in a network, aware only of the class of the customers and the routing that depends on the class, then the network would be functioning with no input from the administrator. Furthermore, it would reach some almost-optimal state.

Also, the set-up within such a black box is very simple: totally distributed, easily scalable, and simple to compute.

Chapter 6

Application

6.1 Introduction

We present here a practical application of the theory that has been developed in the previous chapters.

The application is to implement scheduling in an internet router.

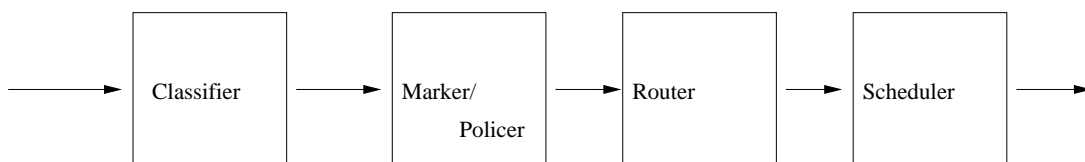


Figure 6.1: Router architecture

6.2 Background

The internet is an ideal example of non-Markovian multiclass network. It has long been a Best-Effort network, meaning that packets sent through the network would receive the

same consideration by a router, whether it is issued by some high priority application or by the lowest priority one. For instance, voice over the internet would receive the same treatment as data, whereas voice is time sensitive, and cannot afford to be delayed. In another example, in case of congestion, packets of any kind are dropped, whereas some might convey information about the congestion, or some might not be as sensitive to packet loss as others. In this case, voice packets can be dropped to some extent, whereas data about say your banking transaction cannot.

This concern is taken into account in the next internet protocol, IPv6, which introduces the notion of differentiation of service. The purpose of this is dual: to guarantee some kind of quality of service to the customers, while at the same time providing the Internet Service Provides (ISP) with some mean of generating more revenue.

The idea is as follows: a client sending packets over the internet would pay a different price for a different guarantee: sending voice packets would be more expensive, but the bandwidth allocated would be guaranteed to some extent.

This opens two areas of investigation: how to price the different levels of service in order to be efficient, which with we are not concerned, and how to schedule the different classes in order to maximize throughput as well as ensure the different guarantees that have been contracted.

There is some congestion control built in the protocols that regulate the internet today. Now, the question of interest is to what extent having different classes would influence the congestion and the utilization of a link over the internet. The main trade-off is that the higher priced levels of service are those that will use a lesser amount of bandwidth, but require that it is available at all times. Namely, there are less customers willing to pay a premium, but some bandwidth needs to be provisioned for them.

The rapid growth of internet traffic makes it difficult to overprovision, so that everybody always finds the needed bandwidth. Hence there is a need for a way of sharing the resource that is compatible with the somehow contradicting goals of fairness and quality of service.

Fairness means that higher quality service cannot totally prevent the service of the other classes.

The mechanism we present is as follows: it is a scheduling policy in a router architecture that ensures a differentiation of service: there is no guarantee as to the absolute level of service, namely the round trip time (RTT) of a packet or the packet drop ratio. There

is only a relative guarantee that ensures that a packet of a given class is indeed given a preferential treatment over a packet from a lower class.

The IETF (Internet Engineering Task Force) proposes a differentiation of service scheme for IPv6 which is as follows:

- A class of service called EF, for Expedite Forwarding, for which customers pay the higher price, but are guaranteed to have the available bandwidth. This class is over-provisioned, and priced accordingly.
- 4 classes of service, denoted AF1 to AF4, for Assured Forwarding, for which the IETF has not defined any scheduling or routing requirement.

This Differentiation of Service is defined in a couple Requests For Comments, (RFCs [58], [7], [37], [36]).

6.3 Application of the Multiplexing Policy to a Router Architecture

The router architecture is composed of several modules:

- an input buffer, that receives the packets from a given link. Each incoming link has its own input buffer.
- a classifier, that reads the IP 5-uplet (source address, source port, destination address, destination port, type of service), and mark the packet accordingly.
- a policer or shaper that shapes the traffic so as to prevent bursts (usually using a token bucket algorithm). It drops packets according to the marking from the classifier.
- a router that looks up the next hop in a routing table, using some routing algorithm (Routing Information Protocol, RIP, or Open Shortest Path First, OSPF, or any other routing algorithm using some cost optimization or some distance measure), and forwards the packet to the corresponding output buffer on the right output link.
- a scheduler that selects which packet from the output buffer is to be sent first on the output link.

It is to be understood that the forwarding and look up in the buffer is done at speeds that are faster than the link capacity. That is that the bottleneck in such a system is not in the routing per se, but in the overflow of a given link.

Our proposed router architecture is independent of the routing mechanism that is chosen. It is also independent of the end-to-end congestion control mechanism.

There are several ways of controlling the congestion over an IP network: the most common ones are UDP (User Data Protocol) that does not do any control, or TCP (Transmission Control Protocol). There are different versions of TCP, with static windows or dynamic windows, slow-start, etc. We enumerate those, but the congestion control that is chosen is of no relevance as far as the router architecture goes: those are issues and protocols that are situated at different layers in the network description.

Usually associated with TCP is the RED (Random Early Detection) algorithm. RED is an algorithm applied at the buffer level that drops packet randomly in order to avoid synchronized windows reduction, and thus avoids the underutilization of a link. This algorithm can also be associated with our proposed router architecture. Our purpose is not to manage congestion, as those devices do, but to try to postpone the time when a link will be congested.

In our architecture, we assume that the output buffer is composed of 5 buffers: each buffer corresponds to a level of service, one for EF, one for AF1, and so on till AF4.

The scheduler does 3 things:

- evaluates the incoming arrival and service rates for each class.
- schedules the tasks according to a coin toss distributed according to the rate information and forward them accordingly.
- reports the congestion of the link, based on the traffic flowing through the link and the respective priorities of this link.

We will now detail each of these steps.

6.3.1 Rate evaluation

The scheduler of each link contains a table with 4 entries per class: the averaged arrival rate, the averaged service time, a congestion indicator, and the time of the last arrival. We detail the estimation algorithm for one class, since it is exactly the same for the other class.

For one class, say for instance AF2, we are given a quadruplet $(\tau(AF2), \sigma(AF2), \rho(AF2), t(AF2))$ corresponding to the interarrival and service time, the congestion indicator and the time of the last arrival to the buffer AF2. Initially, when the link comes up, all these quantities are set to 0.s

Upon arrival of a packet to buffer AF2, the scheduler estimates its service time, that is the time it is going to consume to send it over the link. This is done by acquiring the packet size, and dividing it by the speed of the link.

$$\text{service time} = \frac{\text{packet size (bits)}}{\text{link speed (bps)}} \quad (6.1)$$

It also compute its interarrival time, that is the time between this arrival and the previous arrival to buffer AF2

$$\text{interarrival time} = \text{time} - t(AF2) \quad (6.2)$$

It can then update the interarrival average time and service average time:

$$\begin{aligned} \tau(AF2) &= \gamma_\tau \tau(AF2) + (1 - \gamma_\tau) \text{interarrival time} \\ \sigma(AF2) &= \gamma_\sigma \sigma(AF2) + (1 - \gamma_\sigma) \text{service time} \\ t(AF2) &= \text{time} \end{aligned} \quad (6.3)$$

where γ_τ and γ_σ are numbers in $(0, 1)$. The smaller the γ , the slower the system reacts to any disturbance, but the more accurate the estimate of the average is. On a reliable link, a small value of γ , say a hundredth, would give reasonable estimates.

The estimate of $\tau(AF2)$ and $\sigma(AF2)$ is a one step linear filter, hence we know it converges to the mean. We could use other more evolved filters, the tradeoff being in computation complexity. The advantage of using such a simple procedure is to ensure quick processing, and be consistent with the assumption that the link be the bottleneck, and not the decision making in the router, whether computing the next hop or scheduling the next task.

6.3.2 Scheduling

EF service level: When a packet from the class EF arrives, it has a fixed and higher priority than all the other packets. EF packets are served right away, and have precedence over all other packets. This is necessary to enforce the high guarantee of quality of service imposed on EF packets. On the other hand, EF traffic is limited to a small amount of the available bandwidth. Namely, there is no bursts or congestion, and EF traffic cannot request more than a given amount of the bandwidth. The scheduler does not make any fancy decision for EF traffic, and schedules these packets ahead of everybody.

AF service level: Packets at the AF level compete for the remaining resource after taking EF packets out of the way. AF traffic is not shaped as EF traffic, so it can be bursty, uncontrolled, and requesting all the resource. AF traffic accounts for most of the traffic, hence the necessity of a good scheduling decision.

Using the information provided by the interarrival time, the scheduler can compute the flow for each task as well as the total flow: to do this, it keeps another table with the quantities $(\lambda(AF1), \lambda(AF2), \lambda(AF3), \lambda(AF4), \lambda(\text{link}))$. After each arrival, it updates these quantities:

$$\begin{aligned} \lambda(\text{class}) &= \frac{1}{\tau(\text{class})}, \forall \text{class} \in \{AF1, AF2, AF3, AF4\} \\ \lambda(\text{link}) &= \sum_{\text{class} \in \{AF1, AF2, AF3, AF4\}} \lambda(\text{class}) \end{aligned} \quad (6.4)$$

Best Effort Case It can then update the coin distribution table $(\pi(AF1), \pi(AF2), \pi(AF3), \pi(AF4))$.

$$\pi(\text{class}) = \frac{\lambda(\text{class})}{\lambda(\text{link})} \quad (6.5)$$

and make a scheduling decision by tossing a coin pointing at any class with the probability given by the $\pi(\text{class})$ again and again until it points at a nonempty class.

This is a Best Effort case, in so far as the flow with the most customers is going to be allocated the most bandwidth. This seems reasonable to do in the context of an even pricing of the classes. We have seen in the previous chapter that this Best Effort case would ensure rate stability, and converge to the right distribution of the π 's. Yet, this is incompatible with the economical motivation of the differentiation of service.

Indeed, customers paying more will request less service, but would want to receive a higher priority.

Differentiation of Service. Since customers in class AF1 are likely to be fewer than customers in other classes, but would want a higher service, we fix the value of the $\pi(\text{class})$ so as to ensure a higher quality of service for the higher precedence classes.

We set the π vector to be the norm 1 positive vector that satisfies:

$$\forall i \in \{1, 2, 3\}, \frac{\pi(AF_i + 1)}{\pi(AF_i)} = \gamma_\pi \quad (6.6)$$

where $\gamma_\pi \in (0, 1)$. Setting $\gamma_\pi = 1/2$ for instance would ensure that a class AF1 packet would get twice as much consideration from the scheduler than a class AF2, 4 times as much as AF3, 8 times as much as AF4.

Of course, there is no need to have a geometric distribution like this one, any distribution that satisfies $\pi(AF1) > \pi(AF2) > \pi(AF3) > \pi(AF4)$ would ensure a differentiation of service. The level of differentiation has to be defined by the service agreement between the client and the internet service provider (ISP), as a function of the bandwidth and the price for every class.

The γ 's are parameters that can be defined according to the usage of the router, and fine tuned afterwards. Also the γ 's can be changed dynamically, in a slow start fashion, that has it converging fast, corresponding to a transient situation, then a finite tuning phase, corresponding to the stationary mode.

There is one issue which has to be addressed, and that is the issue of coherence over one ISP's domain of the tools for differentiation of services. Namely that the vector π stays the same at every router. It suffices for one station not running this differentiation of service for the whole scheme to collapse.

6.3.3 Congestion report

The congestion report is a device that indicates the congestion of a link. We can assume that an ISP has a central intelligence that controls the network and requires each router to give some congestion indication so as to optimize the flows in the network. We can also assume that each router is going to forward the congestion indication to the routers it communicates with, and that they would take this into account in their routing tables. We

can also consider the case of a router forwarding the congestion indication to the ingress nodes of the network so as to implement some Call Admission Control (CAC) . What we present here is only a congestion indicator; we are not interested here in the use that is made of this indicator, since it is not necessarily a router issue, but more of a network issue.

We use the quantities $\lambda(\text{class})$ and $\pi(\text{class})$ in order to compute the congestion vector $\rho(\text{class})$. We require that the router stores a vector $\rho(AF1), \rho(AF2), \rho(AF3), \rho(AF4)$ on top of the λ 's and the π 's .

The ρ vector is initialized equal to 0. The algorithm that computes the new congestion indication vector is as follow:

- **3 steps indicator**

This indicator gives a qualitative answer which consists of one of three steps: stable, intermediate and unstable. If we make a comparison, we can say the indicator gives a color, which is green, yellow or red.

In this 3-colors indicator, ρ is then defined as follow

- If for all classes,

$$\lambda(\text{class}) < \frac{\pi(\text{class})}{\pi(AF1)\sigma(AF1) + \pi(AF2)\sigma(AF2) + \dots + \pi(AF4)\sigma(AF4)} \quad (6.7)$$

then $\rho(\text{class}) = 0$ or green, if we prefer the color comparison. We pick 0 in order to insist on the fact that the system is stable.

- If for at least one class, the equation 6.7 is not satisfied, yet the stability condition:

$$\sum_{\text{class} \in \{AF1, \dots, AF4\}} \frac{\sigma(\text{class})}{\tau(\text{class})} < 1 \quad (6.8)$$

then $\rho(\text{class}) = 1$ for all classes, or orange in a color situation. We pick the value 1 which is classically a value of indetermination for the loading of a system to indicate its stability possibly going either way.

- If equation 6.8 is not satisfied, then for all classes, $\rho(\text{class}) = 2$. Once again, we pick the value so as to correspond to the stability of a single queue. If we prefer to use color, then it would correspond to red, since the system is unstable.

The correlation between values of ρ and stability needs not to be explained again, since it comes from the robustness result of the chapter 5

- **Quantitative indicator**

This quantitative indicator function is trying to compute the effect of the cross interaction between classes. Assume for now we only have two classes at one station. If the loading of one class is null, say for class 0, then class 1 does not see any customer from class 0: either there are none, or they zip through so fast that they never delay customers from class 1: class 0 customers are transparent, and they do not influence customers from class 1 at this station.

Remark: class 1 customers still influence class 0 customers in this situation, just a quick look at the Lu-Kumar network should convince the reader that there still is a stability issue.

Yet, the loading of class 1 customer is not affected locally by the class 0 customers: hence we can say that the class 1 load is $\lambda_1\sigma_1$, with obvious notation.

Assume now that will still have two classes at one station, but the loading of class 1 is now equal to 1, and the loading at class 0 is positive. In this situation, class 0 customers will always find class 1 customers, since the system is unstable and the number of customers grows to infinity. Hence, the service time requirement for a class 0 customer is going to be its own service time plus the service time of the customers that the coin toss schedules between the previous class 0 customers and the class 0 customer we are considering.

For such a class 0 customer, we can define a load that is:

$$\rho_e = \lambda_0(\bar{\sigma}_0 + \frac{\pi_1}{\pi_0}\bar{\sigma}_1) \tag{6.9}$$

We denoted this load ρ_e since it is in a way an effective load on the class 0 customers. If the arrival of class 0 customers are spread out and the priority ration $\frac{\pi_1}{\pi_0}$ is such that $\rho_e < 1$, then the class 0 buffer would be stable, whereas the class 1 buffer would overflow really quickly.

If we linearly interpolate between the two values of an *effective* load that we have discussed so far, namely the case of one class bringing no load and the other case of one class saturating the system, we find an indicator for the load and an algorithm to compute it.

This indicator is then validated through computer simulations.

In the next paragraph, we detail the estimation algorithm for the congestion indicator.

Define $\rho(\text{class})$ to be an initially set to the 0 vector.

We define the following iterative step: at each arrival of a new customer, we update the ρ vector as follow (recall that $\lambda(i) = 1/\tau(i)$):

$$\rho(\text{class}) = \lambda(\text{class}) \left(\sigma(\text{class}) + \sum_{i \in \{AF1, \dots, AF4\}, i \neq \text{class}} \frac{\pi(i)}{\pi(\text{class})} \sigma(i) \rho(i) \right) \quad \forall \text{class} \in \{AF1, \dots, AF4\} \quad (6.10)$$

By a Fixed Point Theorem, if all parameters $\tau(i)$ and $\sigma(i)$ converge to a limit, so does $\rho(i)$.

The indicator defines a region in the stability space that does not necessarily correspond to stability. Yet it reflects the behavior of the system.

This concludes the heuristic presentation of a possible application of the rate stability and robustness of the random multiplexing scheduling policy in a router architecture.

6.4 Conclusion

The router architecture we propose is built on the theoretical results from the previous chapters. It uses a sliding window algorithm to evaluate the service rates (mean packet size) and arrival rates, and computes from these a congestion indicator.

Such a congestion indicator could be used in Call Admission Control algorithms, or in a congestion-aware routing protocol. It also schedules the packets in a fair manner, with respect to the QoS requirements, but without starving one class compared to the others. Also, as we mentioned before, it is simple and distributed.

It relies on the robustness result from the previous chapter. Beyond the scope of this thesis is the implementation and validation of such a scheduling policy in an actual IPv6 network. We leave this for future research.

Chapter 7

Conclusion

We have described in this work an extensive account of a scheduling policy of particular interest, since it guarantees some kind of stability in a network.

We also have proved a general result in the case of a general topology network, and this is the main contribution of this dissertation, that for any multiclass network topology there exists a scheduling policy that will stabilize the network. Such a policy should be known by any network manager since it is always suboptimal. Hence, any time a network manager would not know of an optimal policy such that no other policy can improve on, it should use the random multiplexing policy as a safety net.

Chapter 2. We defined a network model that we used in the subsequent chapters, and motivated our work through a review of the literature on the subject.

Chapter 3. In this part, we proved the rate-stability of the random multiplexing policy. Namely, we constructed a distributed, local, intuitive and simple scheduling policy that does not need to have more information on the network topology than the arrival flows of customer to each node, and proved that it would stabilize any system, despite its using so little knowledge about the system.

Chapter 4. In this part, we focused on the Markovian case. We proved some distributional bounds on the waiting time of a customer in our model, when the stochastic sequences involved were i.i.d. and independent of each other. Using then a fluid limit theorem, we were able to prove the convergence of the state space of the system to an equilibrium distribution.

This result is of tremendous importance, since it states that in the Markovian case, the random multiplexing is more than just suboptimal, it is optimal as far as throughput goes.

Chapter 5. In here, we discussed the stability of our stabilizing policy with respect to a change in the arrival rates. In most applications, service rates are given, and their stochasticity does not evolve dramatically with time. Arrival rates on the other hand might be dynamically assigned. We give a variation set on the arrival rates that still ensures the stability, and define a scheduling policy that adapts itself to variations in the arrival rates. This is another result of tremendous importance, since it gives a very robust scheduling policy in most application.

Chapter 6. In this last part, we put the previous results together in order to define a scheduling policy used in a router that ensures differentiation of service as a quality of service device.

There are many topics that need to be investigated in this field of study. A question of dear interest is that of extending the distributional bound result we proved in the i.i.d. case to the more general case of stationary and ergodic sequences.

Appendix A

Appendix

Bibliography

- [1] F. BACCELLI, S. FOSS, *Ergodicity of Jackson-type queueing networks*, (Queueing Systems 17, (1994) pp.5-72).
- [2] F. BACCELLI, P. BREMAUD, *Palm probability and stationary queues* (1987) Springer.
- [3] N. BAMBOS, K. WASSERMAN *Asymptotic optimality of statistical multiplexing in pipelined processing*, (Queueing Systems 21 (1995) 97-123).
- [4] N. BAMBOS *On closed ring queueing networks*, (J.App. Prob. 29, 979-995 (1992)).
- [5] N. BAMBOS, K. WASSERMAN *On stationary tandem queueing networks with job feedback*, (Queueing Systems 15 (1994) 137-164).
- [6] N. BAMBOS, J. WALRAND, *Scheduling and stability aspects of a general class of parallel processing systems*, Adv. Appl. Prob, (1993), vol.25, 176-202.
- [7] S. BLAKE, D. BLACK, M. CARLSON, E. DAVIES, Z. WANG, W. WEISS, *An Architecture for Differentiated Services*, RFC 2475, available at <http://www.ietf.org/rfc/rfc2475.txt>.
- [8] F. BASKETT, K. CHANDY, R. MUNTZ, F. PALACIOS *Open, closed and mixed networks of queues with different classes of customers*, (Journal of the ACM, vol.22, (1975), 248-260).
- [9] D. BERTSIMAS, D. GAMARNIK, J. TSITSIKLIS, *Stability conditions for multiclass fluid queueing networks*, IEEE transactions on automatic control, (1996), vol. 41, 1618-1631.
- [10] A.A. BOROVKOV *Asymptotic Methods in queueing theory* (1984) Wiley.
- [11] A.A. BOROVKOV *Limit theorems for queueing networks. I* Theory Prob. Appl. (1986), vol.31, 413-427.
- [12] M. BRAMSON *Instability of FIFO queueing networks*, (Annals of Applied Probability, 1994, Vol.4, No.2, 414-431).
- [13] M. BRAMSON *Instability of FIFO queueing networks with quick service times*, (Annals of Applied Probability, 1994, Vol.4, No.2, 694-718).
- [14] M. BRAMSON *Convergence to equilibria for fluid models of FIFO queueing networks*. (Queueing Systems: theory and applications, (1996) vol.22, 5-45).
- [15] M. BRAMSON *Convergence to equilibria for fluid models of head-of-the-line proportional processor sharing queueing networks*. (Queueing Systems: theory and applications, (1997) vol. 23, 1-26).

- [16] H. CHEN, *Fluid approximations and stability of multiclass queueing networks: work conserving disciplines*, (Annals of Applied Probability, (1995), vol. 5, 637-665).
- [17] H. CHEN, H. ZHANG, *Stability of multiclass queueing networks under priority service disciplines*, (Operations Research (1998), to appear).
- [18] J. COHEN *The single server queue*,(1982) 2nd ed. New-York, North Holland.
- [19] J.G. DAI *On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models*, (Annals of Applied Probability (1995), vol.5, 49-77).
- [20] J.G. DAI *Stability of open multiclass queueing networks via fluid models*. (In Kelly, F., and Williams, R., editors, Stochastic Networks, vol.71 of the *IMA volumes in mathematics and its application*, New-York, Springer-Verlag (1995), 71-90.
- [21] J.G. DAI, S.P. MEYN, *Stability and convergence of moments for multiclass queueing networks via fluid limit models*, IEEE transactions on automatic control (1995), vol. 40, 1889-1904.
- [22] J.G. DAI, J. VANDEVATE. *The stability of two-station multi-type fluid networks*. Operations Research (1999) to appear.
- [23] J.G. DAI, G. WEISS *Stability and instability of fluid models for re-entrant lines*, Mathematics of Operations Research (1996), vol.21, 115-134.
- [24] J.G. DAI, J.J. HASENBEIN, J.H. VANDE VATE, *Stability of a Three-Station Fluid Network*, Queueing Systems, (1999), to appear.
- [25] D. DOWN, S.P. MEYN *Piecewise linear test functions for stability and instability of queueing networks*, Queueing Systems: theory and applications (1997), vol.27, 205-226.
- [26] D. DOWN, S.P. MEYN *Stability of acyclic multiclass queueing networks*, IEEE Transactions on automatic control (1995), vol.40, no.5, 916-920.
- [27] R. DURRETT, *Probability: Theory and Examples* (Duxbury, (1996)).
- [28] A.K. ERLANG *The theory of probabilities and telephone conversations*. (Nyt Tidsskrift Matematik,(1909) vol. 20, 33-39).
- [29] A.K. ERLANG *Solution of Some Problems in the theory of probabilities of significance in automatic telephone exchanges*, (1917) (English translation: P.O. Elec.Eng.J. (1918) vol.10, 189-197).
- [30] R.K. GETTOOR *Transience and recurrence of Markov processes*. In Azéma and Yor, eds., Séminaire de probabilités XIV, (1979), New-York, Wiley, 397-409.
- [31] D. GROSS, C. HARRIS *Fundamentals of queueing theory*, Wiley.
- [32] J.M. HARRISON, *Brownian models of queueing networks with heterogenous customer populations*, Proceeding IMA workshop on Stochastic Differential Systems (1998), Springer-Verlag.
- [33] J.M. HARRISON, V. NGUYEN, *Brownian models of multiclass queueing networks: current status and open problems*, Queueing systems: theory and applications (1993) vol. 13, 5-40.

- [34] J.M. HARRISON, L.M. WEIN *Scheduling networks of queues: heavy traffic analysis of a simple open network*, Queueing systems, (1989), vol.5, 265-280.
- [35] J.M. HARRISON *Blanced fluid models of multiclass queueing networks: a heavy traffic conjecture*, Stochastic networks, (1998), vol. 71 of the IMA volumes in Math. and Appl. Prob., Kelly and Williams eds.
- [36] J. HEINANEN, F. BAKER, W. WEISS, J. WROCLAWSKI, *Assured Forwarding PHB Group*, RFC 2597, available at <http://www.ietf.org/rfc/rfc2597.txt>.
- [37] V. JACOBSON, K. NICHOLS, K. PODURI, *An Expedited Forwarding PHB*, RFC 2598, available at <http://www.ietf.org/rfc/rfc2598.txt>.
- [38] J.R. JACKSON *Networks fo waiting lines*. Operation Research (1957), vol.5, 518-521.
- [39] J.R. JACKSON *Jobshop-like queueing systems*, Management Science (1963), vol. 10, 131-142.
- [40] F. KELLY *Networks of queues with customers of different types*, (Journal of Applied Probability, (1975), vol. 12, 542-554).
- [41] F. KELLY *Networks of queues*, (Advances in Applied Probability, (1976), vol.8, 416-432).
- [42] f. KELLY *Reversibility and stochastic Networks*, (1979), New-York, Wiley.
- [43] F. KELLY *Loss Networks*, Annals of Applied Probability (1991) vol. 1, 319-378.
- [44] L. KLEINROCK, Queueing systems, vol. 1: Theory (1975) New-York, Wiley.
- [45] L. KLEINROCK, Queueing systems, vol. 2: Computer Applications (1976) New-York, Wiley.
- [46] A.N. KOLMOGOROV, *Sur le problème d'attente*, Mat. Sbornik (1931) vol. 8, 412-436.
- [47] P.R. KUMAR *Re-entrant lines*, (Queueing Systems 13 (1993) 87-110).
- [48] P.R. KUMAR, T.I. SEIDMAN *Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems*, IEEE transactions on automatic control (1990), vol. 35, 289-298.
- [49] P.R. KUMAR, S.P. MEYN *Stability of queueing networks and scheduling policies*, IEEE transactions on automatic control (1995), vol.40, no.2, 251-261.
- [50] S. KUMAR, P.R. KUMAR. *Fluctuation smoothing policies are stable for stochastic re-entrant lines*, Discrete Event Dynamic Systems, (1996) vol.6, 361-370.
- [51] S. KUMAR, P.R. KUMAR, *Performance Bounds for Queueing Networks and Scheduling Policies*, (IEEE Trans. Automatic Control, vol. 39, no.8, august 1994, 1600-11).
- [52] S.H. KUMAR, P.R. KUMAR, *distributed scheduling based on due dates and buffer priorities*, IEEE Transaction on automatic control, (1991), vol. 36, 1406-1416.
- [53] R. LOYNES *The stability of a queue with non- independent inter-arrival and service times*, Proc. Cambridge Phil. Soc. 58, (1962), 497-520.
- [54] C.MAGLARAS, *Dynamic scheduling in multiclass queueing networks: Stability under discrete review policies*, Queueing Systems; theory and applications (1998) submitted.

- [55] C. MAGLARAS, *Discrete-review policies for scheduling stochastic networks: fluid asymptotic optimality* Annals of applied probability (1998) submitted.
- [56] S.P. MEYN *Transience of multiclass queueing networks via fluid limit models* Annals of applied probability (1995), vol. 5, 946-957.
- [57] S.P. MEYN, D. DOWN, *Stability of generalized Jackson networks*, Annals of Applied Probability, 4, (1994), pp.124-148.
- [58] K. NICHOLS, S. BLAKE, F. BAKER, D. BLACK, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, RFC 2474, available at: <http://www.ietf.org/rfc/rfc2474.txt>.
- [59] A.N. RYBKO, A.L. STOLYAR, *Ergodicity of stochastic processes describing the operation of open queueing networks*, Problems of Information Transmission, 228 (1992), pp.199-220.
- [60] T.I. SEIDMAN, *'First Come First Served' can be unstable*, IEEE Transactions on Automatic Control.
- [61] K. SIGMAN, *The stability of open queueing networks*, Stochastic processes and their applications, (1990), vol. 34, 11-25.
- [62] A.L. STOLYAR *On the stability of multiclass queueing networks: a relaxed sufficient condition via limiting fluid processes*. Markov Processes and related fields (1995), 491-512.
- [63] R.J. WILLIAMS *Diffusion approximations for open multiclass queueing networks: sufficient conditions involving state space collapse*. Queueing Systems: theory and applications, (1998), vol.30, 27-88.

Index