MASTER'S DEGREE THESIS

Master of Science in Advanced Mathematics and Mathematical Engineering

Title: A logistic queue model for network traffic modeling and simulation

Author: Franco Coltraro Ianniello

Advisor: Luis Velasco Esteban

Co-Advisor: Marc Ruiz Ramírez

Department: Computers Architecture

Academic year: 2016-2017

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Agradecimientos

Me gustaría agredecer a mis tutores Luis Velasco y Marc Ruiz y en general a todo el Grupo de Comunicaciones Ópticas por haberme dado la excelente oportunidad de trabajar un año entero en un ambiente de investigación emocionante y estimulante. Agradezco tanto el apoyo académico y personal que el grupo me ha brindado, como el económico.

Tampoco puedo dejar de agradecer a mis amigos de Madrid y Barcelona el haberme aguantado un año entero diciendo que *mi cola era la mejor del mundo*. Finalmente, me gustaría recordar el apoyo incondicional de mis padres y mi hermano.

Abstract

In this work we present a continuous queuing model called the *logistic queue model*. We show that in terms of queue size and outflow prediction, our model is as precise as a discrete event simulator with the additional advantage of speed in simulations. We prove mathematically that our proposed model has all the theoretical properties one should expect, e.g. positivity of queue, FIFO property. Finally, in contrast with other continuous models –*Vickrey's point-queue model*– our model can be easily integrated numerically and moreover it allow us to naturally explore multiple extensions relevant to telecommunication networks such as: finite queues, multiple servers, priority queues, etc.

Keywords: queueing theory, fluid flow model, fluid queues, telecommunication networks, simulation, modeling, differential equations.

Resumen

En este trabajo presentamos un modelo continuo de teoría de colas llamado el *modelo de la cola logística*. Mostramos que en cuanto al tamaño de la cola y a la predicción del fujo de salida se refiere, este es tan preciso como un simulador de eventos discretos con la ventaja de ser más rápido en las simulaciones. Demostramos matemáticamente que el modelo posee propiedades teóricas importantes, e.g. positividad de la cola, propiedad FIFO. Asimismo, a diferencia de otros modelos continuos –*modelo de la cola puntual de Vickrey*– nuestro modelo puede integrarse numéricamente con facilidad y además puede extenderse para casos relevantes a las redes de telecomunicaciones: colas finitas, colas con prioridades, etc.

Palabras clave: teoría de colas, colas fluidas, redes de telecomunicaciones, simulación, modelización, ecuaciones diferenciales.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The expected explosion of services offered on the Internet has revealed new paradigms in telecommunication networks. Traffic forecasts in [1] and [2] reveal that the exchange of data between data-centers (DCs) will reach **905 exabytes** per year by 2019. This is induced by an increasing IP traffic that will be mainly coped by video services: 80% of total IP traffic by 2019. Aiming at interconnecting DCs and services, connectivity from the transport network is required to carry **traffic flows** between network nodes, each consisting of a mix of different services.

From the network perspective, transport networks are currently configured with big **static** fat pipes. They are based on capacity over-provisioning aiming at guaranteeing traffic demand and Quality of Service (QoS). *Cloud-ready transport network* [3] was introduced as an architecture to handle a **dynamic** cloud and network interaction. The evolution towards cloud-ready transport networks (telecom cloud) is based on intelligent control architectures and elastic data planes that can satisfy cloud requirements efficiently for both network and cloud operators [5].

The optimization of telecom cloud infrastructures includes network planning and reconfiguration by means of various mathematical and computational techniques [6]. Among them, **dynamic network simulation** is of paramount importance in order to evaluate the performance of new services and applications over the network. To that end, one of the key elements in the simulation is how to **generate** and **model** network traffic. Traffic generation is a useful technique that enables studying and evaluating the network performance through simulation, when real traffic traces are not available. Authors in [7] presented traffic generators to inject realistic traffic flows in telecom cloud simulated systems. Basically, different type of functions like piecewise linear, polynomial or trigonometric are used to model traffic flow average profiles that can be periodical and evolutionary, e.g. incremental. Besides,

traffic is characterized by a random function around that average.

Simulations based on **flow-based models** present interesting features such as optimum efficiency and easiness of traffic parameter characterization. However, the mix of services that could be aggregated into a flow could be too much heterogeneous and variable to allow a good characterization following such a simple approach. Moreover, interesting outputs that could be measured in simulation such as end-to-end latency or node switching delay cannot be accurately obtained due to the inherent nature of flow-based models that hides any individual service behavior. This is due to the fact that this approach models traffic directly without considering **queues or servers** in the network.

An alternative approach is to use a packet based simulations where traffic is generated and simulated at the packet level, i.e. characterizing the packet generation of an individual service and running a **discrete event simulation** that process the transition of the packet from source to destination to every intermediate node [8]. This way we do consider queues and servers in the network, and hence can measure accurately quantities of interest such as latencies or delays. It is worth noting that the amount of information that could be obtained from a discrete event simulation based on a packet-based traffic generator is unbeatable. However, when high income bit-rates e.g. in the order of dozens of Gb/s per flow, the **computational cost** of processing the resulting huge number of packets is prohibitive even for small networks. To this aim, in the last years several research works have focused on providing alternative simulation environments allowing a fine granular view of the system comparable with that of packet-based simulations with that much more efficiency and **scalability** of flow-based ones.

One of the most successful approaches for doing so has been the use of **fluid flow models** [9]. Basically the idea is to consider only changes in rates of traffic flows. This can result in large performance advantages because information about the individual packets is not considered but only their joint behavior. A fluid simulator records the changes in the fluid rate in the source and the queue, while a packet simulator records the events of all the packets in the system. The abstraction takes place when packet flows with little *time slots* separations are considered to be in the same fluid flow with a constant fluid rate. Little time variations among packets are not considered, and in this way the number of events is reduced. Of course, a critical issue of this kind of models is how to choose the time slots when one is given an arbitrary flow. Moreover, recording the changes in the fluid rate can still be seen as being discrete in nature.

In the end, that type of models can be seen as discrete variations of the famous *Vickrey's point-queue model* [11]. This is a purely *continuous* model –we do not have to keep track of time slots– of a queue with a server formulated through an **ordinary differential equation** (ODE). It has been throughly studied, being one of the most notable works the one done in [12]. In that article the authors find an explicit solution to the previous differential equation and with that formula at hand they are able to prove many desirable properties of the point-queue model, e.g. positivity of the queue size and the first in, first out (FIFO) property.

However there are some issues with the model. The most obvious one is that the vector field induced by the differential equation (see 3.4) is not continuous and hence in general there do not exist classical solutions. This is computationally and numerically problematic since we can not apply usual ODE integrators. Nevertheless it is worth noting that the authors give in [13] an applicable numerical algorithm. The main problem that remains is generalization. Since the numerical algorithm they derive relies in the exact solution they found, we can not expect to have numerical schemes for more general cases of interest in telecommunication networks such as: finite queues or priority queues.

1.2 Contributions

In this work we present a novel queuing model to solve the aforementioned problems. This new model, which we call the **logistic queue model** can be seen as a smooth formulation of the point-queue model. This is important, since in such a way we are able to use common numerical integrators. We give original mathematical proofs of all the **theoretical properties** one should expect: i) *positivity* of the queue size, ii) *asymptotic behavior*: the queue gets empty if the inflow does not overflow and iii) *FIFO property*: the system satisfies a first in, first out discipline.

Moreover, we validate the model comparing it with a discrete event simulator. This way we show that for many purposes our model is as **precise** as a discrete one with the advantage of speed in simulations. We compare simulation times and conclude that the logistic queue model is several orders of magnitude **faster** than a discrete one. Finally, in contrast with the point-queue model, our model allow us to easily explore multiple **extensions** –important for telecommunications– to more general scenarios such as: finite queues, multiple servers, priority queues, etc.

Lastly we give an **application** of the model in a telecommunication network where we confirm its versatility and potential.

1.3 Organization

The rest of this work is organized as follows: in chapter 2 we review some basic concepts of optical telecommunication networks and queuing systems. In chapter 3 we present the logistic queue model. There, we give a full derivation of it, with all the motivations behind the equations. Then we present original proofs of theoretical properties of the model. Finally we end the chapter discussing some issues related with the application of the model with *real data*. In chapter 4 we present important extensions of the model so that it can handle more general situations common in telecommunication networks. In chapter 5 we validate the model comparing its performance with a discrete event simulator. Finally in chapter

6 we give an application of the model to show its potential. The report ends with some concluding remarks.

Chapter 2

Background

In this chapter we review some basic concepts of optical networks and queuing systems.

2.1 Introduction to dynamic optical networks

An optical network can be defined as a graph with its representative equipment based on a certain optical technology. In general, it is represented by an **undirected graph** where the edges are fiber optic links and the vertices are optical nodes capable of establishing and deleting optical connections. The optical technology uses a range of frequencies of the total Optical Spectrum (OS), and it is measured in Gigahertz (GHz). The capacity of an optical link depends on the OS width and other factors like the spectral efficiency of established connections. This part of the network is called the **optical layer**.

On top of the above described optical layer, large packet nodes (e.g. IP routers or Ethernet switches) are collocated, they serve as end points of network traffic, as well as support intermediate transit operation. Thus, a **traffic flow** is a request of bitrate to be transported between a source packet node and a termination packet node usually expressed in Megabits per second (Mb/s) or Gigabits per second (Gb/s). When a demand is accepted, an optical connection in the network is established between source and termination nodes; these optical connections are called lightpaths since they allow the data transmission as a light wave.

From the abstracted view of the packet layer, a lightpath is considered as a **virtual link** directly connecting two packet nodes. In this layer, data is transmitted discretely in the form of **packets**. Packet sizes are usually measured in bytes (1 byte is equal to 8 bits). Thus, a virtual topology is used to route and serve traffic between source and destination nodes. Hence we have two layers in the network: the *physical* layer –the optical layer– where data is transmitted as a light wave and a *virtual* layer –the packet layer– where data is transmitted in the forms of packets.

In the packet layer is where we have queues and servers. There is where our main focus will be for the rest of this report. Packets are sent from a source packet node to a termination packet node using **servers** of given velocities (usually in Gb/s). When a packet finds a server occupied it enters into the waiting line, i.e. a **queue**, until the server becomes free and it can be used.

Two different approaches can be considered for planning and operating communications networks: static and dynamic traffic scenarios. In **static** traffic scenarios, no changes are considered in the connections established during the working period of the network. Thus, the information about client demands to be served, i.e. the source and destination nodes and the required bandwidth, is known in advance. Since the routing of those demands can be computed before the network begins to operate and no changes are allowed during the working time, the optimality of the network planning is always kept.

On the contrary, in **dynamic** traffic scenarios demands are not known in advance and connections are continuously set up and torn down. We assume that client requests arrive to the network following a certain probability distribution function. Moreover, connections remain active during a certain period of time, i.e. the service time, which can be also modeled by another probability function. In this scenario is where **simulation** and **modeling** come into play. Given a particular situation where we model client requests, network capacities (queues lengths, server speeds) network topology, etc; we can simulate the network and evaluate its performance.

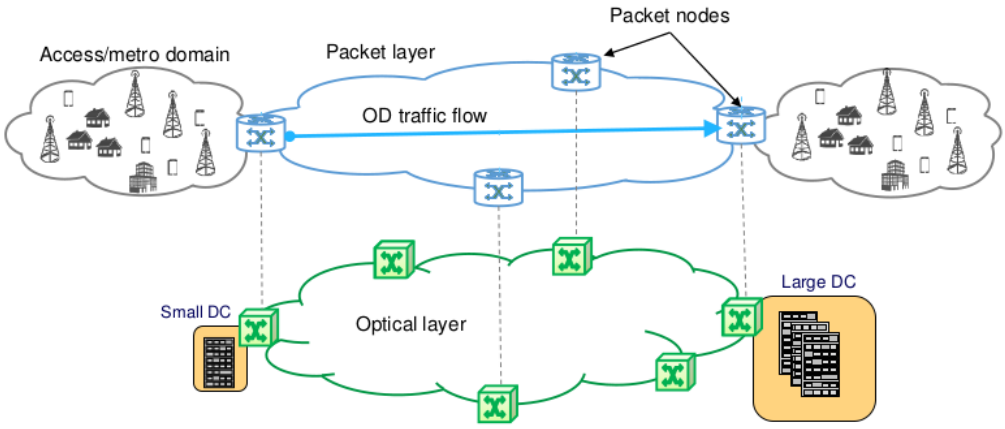

Figure 2.1: Simplified network architecture example.

A simplified **network architecture** is presented in Figure 2.1, where an optical transport network containing a set of packet nodes is interconnected by means of virtual links in the packet layer. Each virtual link is supported by one or more optical connections in the optical layer. Thus, traffic is served through such capacity.

The network in the example interconnects different access and metropolitan areas where

service end users are. All traffic generated in one area targeting other area in the operator domain or another network (e.g. Internet) is aggregated in the source packet node and sent towards the destination node, thus becoming a traffic flow. For instance, in the example above, the depicted OD flow can include at the same time: i) traffic between end users of both areas, ii) traffic between the source area and a large Data Center (DC), and iii) traffic from the small to the large DC.

2.2 Queuing models

Queueing theory is the study of waiting lines. Using mathematical and computational tools a model is constructed so that queue lengths and waiting times can be predicted. In general there exist many types of queueing models, being the the **probabilistic** [14] the most common ones. They are very important conceptually and theoretically, nevertheless we will not make use of them explicitly.

The second most usual models are **discrete event simulation** systems. Before describing in some detail how we can model a queue and a server using a discrete event simulator we will describe the functioning of our system in more intuitive terms.

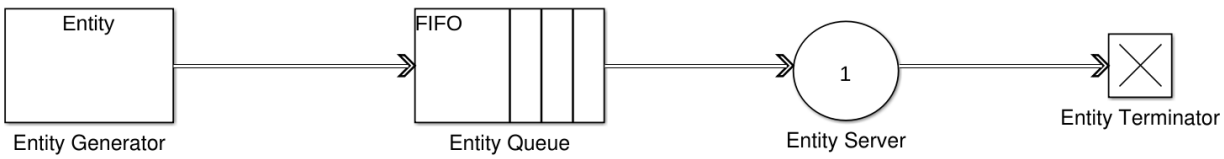

Figure 2.2: Schematic picture of a system consisting of a server with a queue.

In Figure 2.2 we show a schematic picture that models a server with a queue. Entities (e.g. customers, packets) are generated from *Generator* following some probability distribution (e.g. exponential). Each entity is created at a specific instant of time. Then it has to be served, if the *Server* is free when it was created, it is processed, else it joins the *Queue*. We assume that the server has an associated speed μ, this means that our system is capable of processing μ entities per unit of time. When the server gets free, the first entity in queue goes to the server and abandons the waiting line. This type of queues are called first in, first out (FIFO). Finally all entities end in a *Sink* where they are terminated.

Now we can describe the *flow-chart* associated to a discrete event simulator in more precise terms. A discrete event simulator consists of an ordered **list of future events** to be

executed. Each event has an associated instant of time when it has to be done.

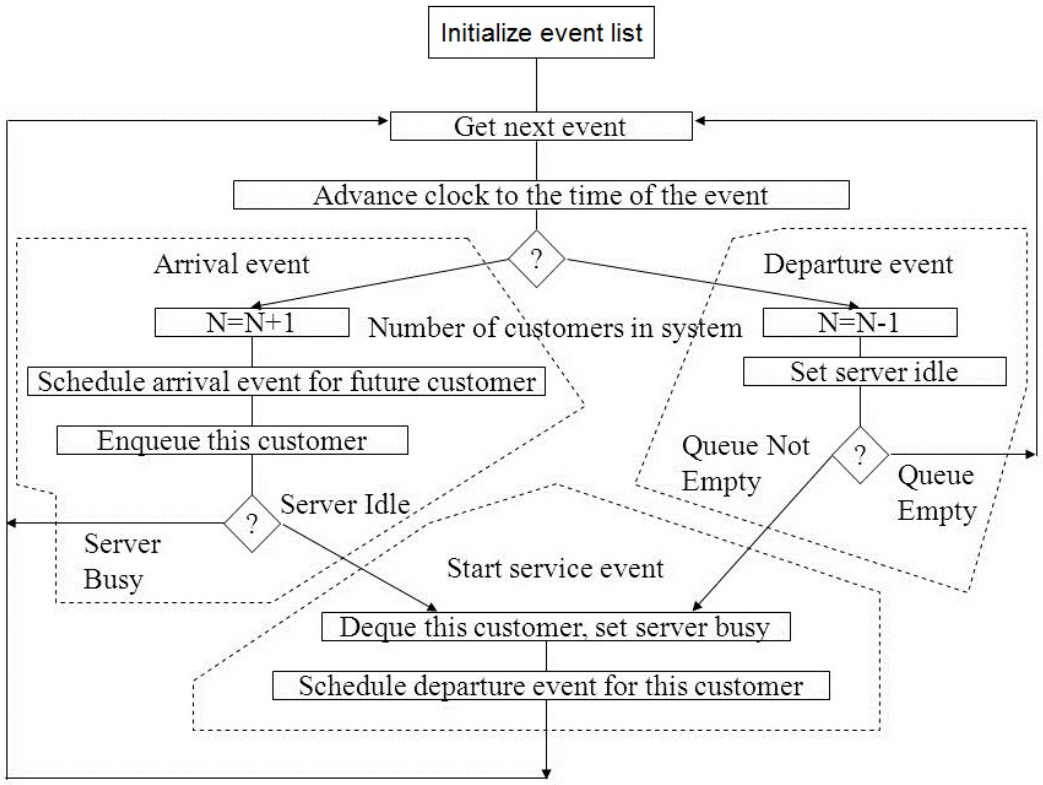

Figure 2.3: Flow-chart associated to system consisting of a server with a queue.

The simulator only considers instants at which an event occurs, in the meantime everything remains unchanged. Internally a **simulation clock** is used to keep track of simulated time. To process an event in the list, we advance the clock to the time of the event, then we change the system state to reflect the impact of the event. For example, if a customer arrives it must be enqueued. Also, new events caused by the event at hand are scheduled and put in the list of future events. For example, if the server is idle, it starts to serve the customer, so a new event has to be generated indicating the time at which the server will be empty again. The simulation ends when no events are left in the list. A detailed scheme of the whole process for our system is shown in Figure 2.3.

Lastly, we also have **fluid flow models** where we allow arrivals to be continuous rather than discrete. This means that instead of considering each entity discretely, we consider the continuous flow generated by them. Thus, the queue size becomes a continuous function of time. The logistic queue model is of this type. In the next chapter we will see how they

come up naturally as a consequence of a conservation law.

2.3 Summary

In this chapter we have given the basic concepts of optical dynamical networks relevant for the rest of the report. The main idea is that data is transmitted discretely in the form of packets using servers from a source node to a destination node in the network. Moreover we have explained in some detail how we can use a discrete event simulator to model a server with a queue. This is an important model because we will compare the performance of the logistic queue model with it. It is important to stress that in contrast to a discrete event simulator, the logistic queue model is a fluid flow model, which means that it does not consider each packet independently but the flow generated by them. This will be the topic of next chapter.

Chapter 3

The logistic queue model

In this chapter we present the logistic queue model. We give a full derivation of it, with all the motivations behind the equations. Then we give original proofs of theoretical properties of the model. Finally we end the chapter discussing some issues related with the application of the model in a realistic data scenario.

3.1 Derivation

Assume we have a system with an inflow of entities arriving to a server of constant speed μ with an infinitely long queue as the one of Figure 3.1. In probabilistic terminology the system is denoted as $G/D/1$ [14]. We will derive an equation that relates the number of entities in queue C with the inflow to the system f and the outflow g.

Figure 3.1: Example of a system with a queue and a server of speed μ.

Note that each quantity has the following units:

- $[\mu]$ = entities/time.
- $[C]$ = entities.
- $[f] = [g]$ = entities/time.

We will assume that the amount of entities in queue $C(t)$ in the instant t is a real number. This will be a reasonable approximation if the inflow is large enough: $f \gg 1$, physically this means that *many* entities arrive to the system (which is the case in telecommunication networks). Also, this will make us not to distinguish between entities in queue and entities in the system. Therefore, for a short period of time dt the amount of entities that have arrived to the system is approximately $f(t) \cdot dt$ and likewise the number of them that have abandoned it is $g(t) \cdot dt$. This gives us the following **conservation law**:

$$C(t + dt) = C(t) + f(t) \cdot dt - g(t) \cdot dt,$$

which in the limit when $dt \to 0$ reads:

$$C'(t) = f(t) - g(t). \tag{3.1}$$

We shall assume that there is a **functional relationship** between f, g and C, and express the outflow as a function of the inflow and the queue size:

$$g(t) = g(f, C) = \mu + e^{-\alpha C(t)} \left[\min\{\mu, f(t)\} - \mu\right]. \tag{3.2}$$

where:

1. $\mu > 0$ is maximum outflow rate of the server.
2. α is a positive parameter taken to be $\frac{\rho}{\mu}$, where $\rho = \frac{\lambda}{\mu}$ is the average intensity, λ is the mean of the inflow to the system:

   $$\lambda = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} f(t)dt$$

   and $[t_0, t_1]$ is the interval of definition of f.

The rationale behind this definition of $\alpha = \frac{\rho}{\mu}$ is the following: dividing by μ normalizes the queue size, and weighting by ρ takes into account the intensity of the inflow. Note that by construction we have:

1. If at t the queue is empty, i.e. $C(t) = 0$, then the outflow is just $\min\{\mu, f(t)\}$.
2. If the inflow is bigger than the maximum capacity, i.e. $f(t) > \mu$, then the outflow is just μ.
3. If $C(t) \to +\infty$ then the outflow tends to μ.

It is instructive to have a qualitative idea of how the outflow depends of the queue size. In Figure 3.2 we make a plot of $g = g(f_0, C_0)$ for a fixed time t_0 and varying $C_0 = C(t_0)$.

In the picture we are assuming that $f(t_0) = 0.5$. It is important to stress that this figure reflects *all possible* outflows we may get at a fixed instant of time t_0 depending on the queue size C_0 we have. Finally we put together (3.1) and (3.2) in the following definition:

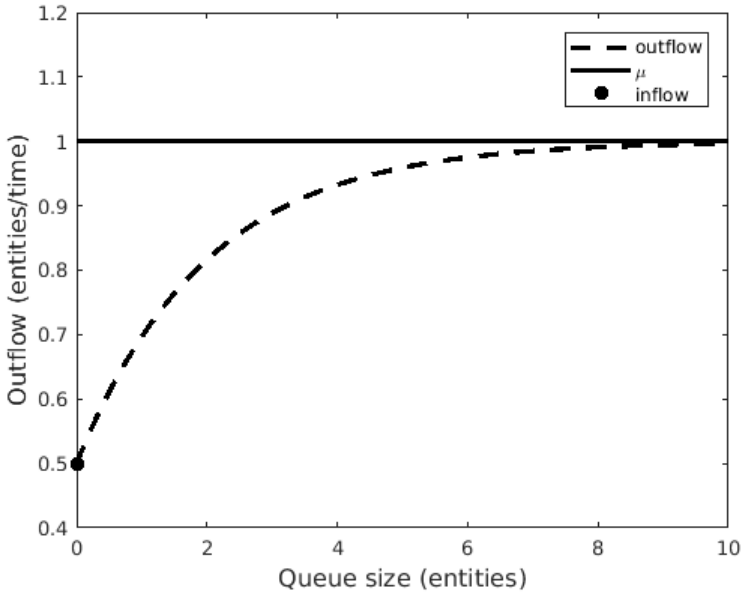

Figure 3.2: Possible outflows we may get at a fixed instant of time t_0 depending on the queue size C_0.

Definition 3.1.1 (**Logistic Queue Model**). Given an initial condition $C(t_0) = C_0$, a continuous and positive inflow function $f : [t_0, +\infty) \to [0, +\infty)$ and a maximum outflow rate $\mu > 0$, the *logistic queue model* is the following ordinary differential equation:

$$\begin{cases} C'(t) &= f(t) - \left[\mu + e^{-\alpha C(t)} \left(\min\{\mu, f(t)\} - \mu\right)\right] \text{ for } t > t_0, \\ C(t_0) &= C_0. \end{cases} \tag{3.3}$$

Remark 3.1.2 Observe that if we let $\alpha \to +\infty$ we get the famous *Vickrey's point-queue model* [11, 12]:

$$C'(t) = f(t) - \begin{cases} \min\{\mu, f(t)\} & \text{if } C(t) = 0, \\ \mu & \text{if } C(t) \neq 0. \end{cases} \tag{3.4}$$

Note that the right hand side of the ODE is not continuous when $C \to 0$. In the next section we will prove that the logistic queue model has all theoretical properties of the point-queue model discussed before with the additional advantage that it is easy to integrate numerically because the ODE is smooth. Moreover, we will see that our model allow us to explore multiple extensions to more general scenarios such as: finite queues, multiple servers, priority queues, etc.

3.2 Theoretical Properties

In this section we prove theoretical properties of the model. Let's start with existence, uniqueness and positivity.

Proposition 3.2.1 (**Existence, Uniqueness and Positivity**). *Given an initial condition $C(t_0) = C_0 \geq 0$, a continuous and positive inflow function $f \geq 0$ and a maximum outflow rate $\mu > 0$, we have that there exist a unique continuous differentiable solution $C(t)$ to the system (3.3) defined in an interval $[t_0, T_{max})$ for $T_{max} \leq +\infty$. Moreover such solution is always greater or equal than zero.*

Proof. Existence, uniqueness and differentiability of $C(t)$ in an interval $[t_0, T_{max})$ for $T_{max} \leq +\infty$ is an easy consequence of the fact that the right hand side of (3.3) is a smooth function of the variable C and it is continuous in t, hence we can apply *Cauchy-Lipschitz theorem* [15]. Let us prove now positivity. Assume that for some time $t_- > t_0$ we have that $C(t_-) < 0$. Then, since $C(t)$ is continuous and $C_0 \geq 0$, there must be a t_+ and a t_* such that $t_+ < t_* < t_-$ where $C(t_+) \geq 0$ and $C(t_*) = 0$ (see Figure 3.3). Also, after reducing the interval $I = [t_+, t_-]$ if necessary, we may assume that $C(t)$ is strictly decreasing there. Therefore we have:

$$C'(t) \leq 0 \text{ in } I = [t_+, t_-].$$

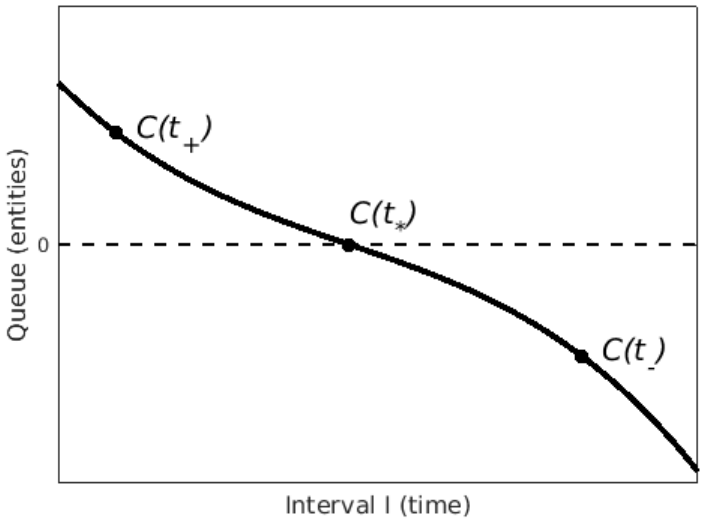

Figure 3.3: C is decreasing in $[t_+, t_-]$.

Let's see how this implies that $f(t) \leq \mu$ in I. Indeed if $f(t) > \mu$ for some $t \in I$ we would have using (3.3) that $C'(t) = f(t) - \mu > 0$, which is a contradiction. Hence we deduce that

$\tilde{C}(t) \equiv 0$ is a solution of

$$\begin{cases} Q'(t) & = f(t) - \left[\mu + e^{-\alpha Q(t)} \left(\min\{\mu, f(t)\} - \mu\right)\right] \text{ for } t \in I, \\ Q(t_*) & = 0, \end{cases} \tag{3.5}$$

which is impossible because we would have two different solutions C and $\tilde{C}$ of (3.5) in I. □

Remark 3.2.2 As a consequence of the previous proposition we deduce that if the queue starts empty, i.e. $C(t_0) = 0$, and the inflow is always smaller than the maximum outflow rate:

$$f(t) < \mu \text{ for all } t \geq t_0,$$

then the queue remains empty for all times and the outflow is equal to the inflow.

Now we are ready to prove some asymptotic properties of the queue:

Proposition 3.2.3 (**Asymptotic behavior**). *Let $C(t_0) = C_0 \geq 0$ be an initial condition, $f \geq 0$ a continuous and positive inflow function and $\mu > 0$ a maximum outflow rate. Then the solution of (3.3) is defined for all times, i.e. it is defined in the interval $[t_0, +\infty)$. Moreover if the inflow satisfies:*

$$f(t) \leq f_\infty < \mu \text{ for all } t \geq t_f \tag{3.6}$$

for some time $t_f > t_0$ then

$$\lim_{t \to +\infty} C(t) = 0.$$

Proof. Note that we can rewrite (3.3) as:

$$C'(t) = f(t) - \mu + e^{-\alpha C(t)} \left(\mu - \min\{\mu, f(t)\}\right), \tag{3.7}$$

but since $C(t) \geq 0$ we have that $e^{-\alpha C(t)} \leq 1$, and moreover $\mu - \min\{\mu, f(t)\} \leq \mu$. Hence we deduce that:

$$C'(t) \leq f(t) - \mu + \mu = f(t),$$

so finally we get that:

$$C(t) \leq C_0 + \int_{t_0}^{t} f(s)ds.$$

Therefore $C(t)$ is defined for all times since f is continuous and hence integrable. Assume now that f satisfies (3.6). Then, using (3.7), we get:

$$\begin{aligned} C'(t) &= (1 - e^{-\alpha C(t)}) \cdot (f(t) - \mu) \\ &\leq (1 - e^{-\alpha C(t)}) \cdot (f_\infty - \mu), \quad \text{for all } t \geq t_f. \end{aligned}$$

Now, it is well known that the exponential satisfies:

$$e^{\alpha C} = \sum_{n=0}^{\infty} \frac{(\alpha C)^n}{n!} \geq 1 + \alpha C, \quad \text{for all } C \geq 0.$$

Hence,

$$e^{-\alpha C} \leq \frac{1}{1+\alpha C} \Leftrightarrow 1 - e^{-\alpha C} \geq 1 - \frac{1}{1+\alpha C} = \frac{\alpha C}{1+\alpha C}.$$

Now, since $(f_\infty - \mu) < 0$ we get the following inequality:

$$C' \leq \frac{\alpha C \cdot (f_\infty - \mu)}{1+\alpha C}.$$

Calling $\beta = \mu - f_\infty > 0$ and rearranging terms we obtain:

$$(1 + \frac{1}{\alpha C}) \cdot C' \leq -\beta \Leftrightarrow \frac{d}{dt}\left(C + \frac{\log C}{\alpha}\right) \leq \frac{d}{dt}\left(-\beta t\right).$$

Therefore, integrating from t_f to t,

$$C(t) + \frac{\log C(t)}{\alpha} \leq -\beta t + k, \quad \text{for all } t \geq t_f$$

where $k = C_f + \frac{\log C_f}{\alpha} + \beta t_f$ and $C_f = C(t_f)$. Applying the exponential in both sides of the inequality one gets:

$$C^{1/\alpha} \cdot e^{C} \leq e^{-\beta t + k}. \tag{3.8}$$

Finally, last equation implies that

$$C(t) \leq e^{\alpha(k - \beta t)}, \quad \text{for all } t \geq t_f.$$

So the queue gets empty exponentially fast as stated. □

Remark 3.2.4 Equation (3.8) gives us an estimate of how fast the queue goes to zero. Assuming (3.6) and given a fixed tolerance $\epsilon > 0$, if we impose that:

$$C(t) \leq e^{\alpha(k-\beta t)} \cdot e^{-\alpha C(t)} \leq \epsilon,$$

then we find that the **emptying time** $T_\epsilon(C_f) \geq t_f$ needed to empty the queue within a tolerance $\epsilon > 0$ satisfies the following bound:

$$T_\epsilon(C_f) \leq t_f + \frac{C_f - \epsilon}{\mu - f_\infty} + \frac{1}{\alpha \cdot (\mu - f_\infty)} \cdot \log\left(\frac{C_f}{\epsilon}\right). \tag{3.9}$$

Note that the previous equation has the following expected *physical* properties:

1. The fastest way to empty the queue is by stopping the inflow, i.e. taking $f_\infty = 0$. On the other hand, if $f_\infty \to \mu$ then the bound goes to infinity as expected.
2. If $\alpha \to +\infty$ then we get that the bound is equal to the emptying time of the point-queue model (3.4).
3. The bound is optimal because if we take $\epsilon = C_f$ then we deduce that $T_\epsilon(C_f) = t_f$.

■

Example 3.2.5 **Constant inflow**. Assume that the inflow to the system is constant, i.e. $f(t) \equiv f_\infty < \mu$. Fix also some $C_0 > 0$ and $t_0 = 0$. Then we have that (3.3) reduces to:

$$\begin{cases} C'(t) &= f_\infty - \left[\mu + e^{-\alpha C(t)}\left(f_\infty - \mu\right)\right] \text{ for } t > 0, \\ C(0) &= C_0. \end{cases} \tag{3.10}$$

This can be easily integrated, so we get the following explicit formula:

$$C(t) = \frac{1}{\alpha} \log\left(1 + ke^{\gamma t}\right) \tag{3.11}$$

where $\gamma = \alpha \cdot (f_\infty - \mu) < 0$ and $k = e^{\alpha C_0} - 1 > 0$. From the formula above is clear that $C(t) \geq 0$ and that $\lim_{t\to+\infty} C(t) = 0$ as expected (see Figure 3.4).

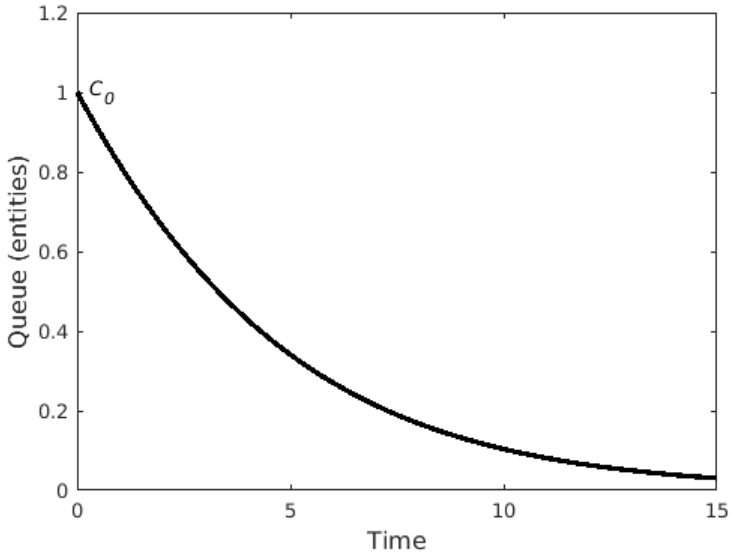

Figure 3.4: Plot of queue size (3.11) for constant inflow $f(t) \equiv 0.5$ with $\mu = 1$, $\alpha = \frac{1}{2}$ and $C_0 = 1$.

Let's now compare the bound we got in previous remark for the emptying time:

$$\hat{T}_\epsilon(C_0) = \frac{C_0 - \epsilon}{\mu - f_\infty} + \frac{1}{\alpha \cdot (\mu - f_\infty)} \cdot \log\left(\frac{C_0}{\epsilon}\right), \tag{3.12}$$

with the exact one we get with the analytical solution:

$$T_\epsilon(C_0) = \frac{1}{\alpha \cdot (\mu - f_\infty)} \cdot \log\left(\frac{e^{\alpha C_0} - 1}{e^{\alpha\epsilon} - 1}\right). \tag{3.13}$$

Note that we have the following:

1. $T_\epsilon(C_0) \leq \hat{T}_\epsilon(C_0)$ as we already know.
2. In the limit:
$$\lim_{\alpha \to +\infty} T_\epsilon(C_0) = \frac{C_0 - \epsilon}{\mu - f_\infty}.$$
So we recover the emptying time of the *point-queue model*.
3. When both C_0 and ϵ are small, we have that $\frac{e^{\alpha C_0}-1}{e^{\alpha\epsilon}-1} \approx \frac{C_0}{\epsilon}$. Hence T and $\hat{T}$ only differ by the linear term $\frac{C_0-\epsilon}{\mu-f_\infty}$.

To conclude this example in Figure 3.5 we make a plot of T and $\hat{T}$ with respect to C_0 and for $\mu = 1$, $f_\infty = 0.5$, $\alpha = \frac{1}{2}$ and $\epsilon = 0.05$.

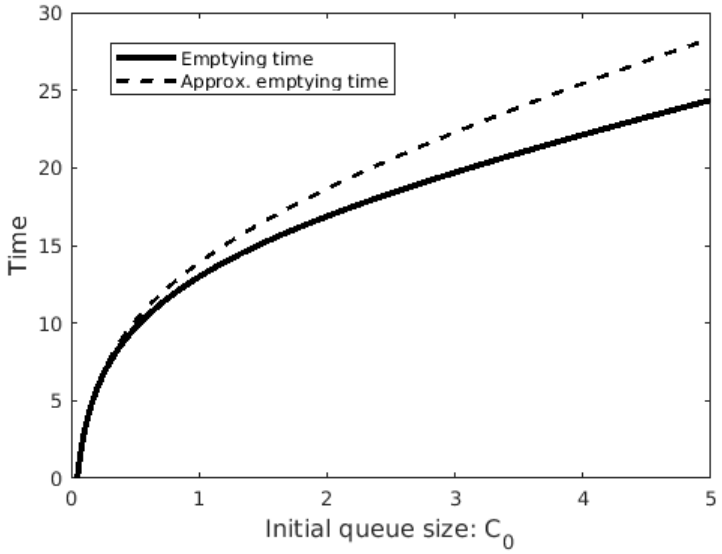

Figure 3.5: Comparison of T_ϵ and $\hat{T}_\epsilon$ for $\mu = 1$, $f_\infty = 0.5$, $\alpha = \frac{1}{2}$ and $\epsilon = 0.05$.

As already said, we can see in the plot that the bound is pretty accurate when C_0 is small, afterwards it deteriorates. ■

To end this section we will show that our model satisfies the FIFO (first-in first-out) property, this is, that the entity that arrives first to the queue is also the one that gets out first. To accomplish this, we introduce the **exit time** of an entity that has arrived in the instant t as:

$$\Lambda(t) = t + \frac{C(t)}{\mu}. \tag{3.14}$$

The rationale behind this definition is the following: $C(t)/\mu$ has units of time and it represents approximately the amount of time needed for emptying the queue if no more entities arrived after t.

We have the following proposition:

Proposition 3.2.6 (**FIFO property**). *Let $C(t_0) = C_0 \geq 0$ be an initial condition, $f \geq 0$ a continuous and positive inflow function and $\mu > 0$ a maximum outflow rate. Then*

$$\textit{If } t < s \textit{ we have that } \Lambda(t) < \Lambda(s).$$

In other words the exit time of an entity that has arrived in the instant t is smaller than other that arrives at $s > t$.

Proof. We have that $\Lambda(t) < \Lambda(s)$ is equivalent to:

$$\frac{C(s) - C(t)}{s - t} > -\mu. \tag{3.15}$$

Since C is differentiable, by the *Mean-Value theorem* [17] we know that there exists a $\xi \in (t, s)$ such that

$$C'(\xi) = \frac{C(s) - C(t)}{s - t}.$$

Now, we distinguish two cases:

1. $f(\xi) < \mu$. There are two sub-possibilities:
   - $C(\xi) = 0$. In this case $C'(\xi) = f(\xi) - f(\xi) = 0 > -\mu$, so (3.15) holds.
   - $C(\xi) > 0$. In this case the outflow g satisfies $0 < g(\xi) < \mu$, hence
   $$C'(\xi) = f(\xi) - g(\xi) \geq -g(\xi) > -\mu.$$
2. $f(\xi) \geq \mu$. In this case $C'(\xi) = f(\xi) - \mu > -\mu$, so (3.15) holds.

□

3.3 Applying the model

When we apply the logistic model with *real data* there are several considerations that must be taken into account. We discuss some of them in the following.

3.3.1 Discrete inflow

The inflow must be given to the logistic model in a continuous fashion. However in practice we only see discrete entities arriving to the system. Hence we need to approximate our ideal continuous inflow. In order to do this, we simply count how many entities have occurred each dt seconds and then add them. In more mathematical terms, in a point of the form $t_{i+1} = (i+1) \cdot dt$ we put

$$f(t_{i+1}) \approx \frac{1}{dt} \sum (\text{entities occurring in } [i \cdot dt, (i+1) \cdot dt]) .$$

It may also be the case that we are already given the inflow at a discrete set of times, in that case we do not need to deal with entities at all.

3.3.2 Integrating the ODE

Next we will need to solve the ODE (3.3). In order to do this we can use very standard mathematical algorithms for integrating differential equations. For example Runge-Kutta methods [16]. The only caution one needs to take care of is the interpolation of the inflow between two successive points t_i and t_{i+1}. A linear interpolation usually suffices.

3.3.3 Queue error due to aggregation time

Approximating the inflow as discussed introduces an error in the model that we can not avoid but which we can estimate. Denote by $C_{log}(t)$ the queue size given by the logistic queue model, and by $C_{disc}(t)$ the real queue size. An entity that enters to the system at $t \in [i \cdot dt, (i+1) \cdot dt]$ will experience a delay of $C_{disc}(t)/\mu$ seconds, nevertheless aggregating each dt seconds the model can not account the fact that the packet got in at t, hence:

$$|\frac{C_{disc}(t)}{\mu} - \frac{C_{log}(t)}{\mu}| \leq dt.$$

If the traffic is really intense, i.e. if $\rho = \frac{\lambda}{\mu}$ is near one, the best we can expect for is:

$$|C_{disc}(t) - C_{log}(t)| \sim \mu \cdot (1-\rho) \cdot dt. \tag{3.16}$$

In chapter 5 we verify empirically previous equation.

3.4 Summary

In this chapter we have presented the logistic queue model and its main theoretical properties. We have given original mathematical proofs of reasonable properties all queueing models ought have: positivity of the queue, emptying of the queue and FIFO property. We have seen how fluid flow models come up as a consequence of a conservation law. Then the logistic queue model appears as a functional relationship of the queue size, the inflow and the outflow. We have also discussed issues related to the application of the model with real data. In the next chapter we will discuss extensions of the basic model which will turn out to be useful in the framework of telecommunication networks.

Chapter 4

Extensions for Communication Networks

In this chapter we present variations of the model that allow us to address more general scenarios present in telecommunication networks than the one assumed until now. Namely:

- If we have a probabilistic distribution on service times, i.e. a G/G/1 system.
- If we have a finite queue.
- If we have k servers, i.e. a G/D/k system.
- If two flows join in a server and then they separate following different paths.
- If we have priority queues.

Note that we may also have combinations of the previous cases.

4.1 Variable service times

Until this point we have assumed that μ is constant. Nevertheless it is very easy to see that all results that we have proven are still valid even when $\mu = \mu(t)$ depends on time as long as $\mu(t) \geq 0$ for all t.

4.2 Finite queue

Assume we are only able to store a finite number of entities $C_{max} > 0$ in queue. We would like to add this restriction to the model. The idea is to annihilate the inflow when the queue

size overflows the maximum storage capacity. Ideally we replace f for $\tilde{f}$ where:

$$\tilde{f}(C(t), t) = H(C(t)) \cdot f(t)$$

and H is a *Heaviside function*:

$$H(C) = \begin{cases} 1 & \text{for } C \leq C_{max}, \\ 0 & \text{for } C > C_{max}. \end{cases}$$

The only problem with this approach is that we would be introducing discontinuities to C'

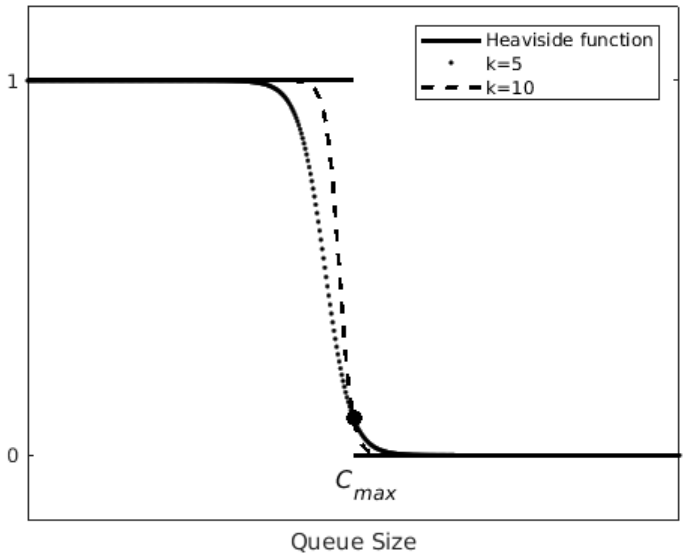

Figure 4.1: Approximations of *Heaviside function* for $k = 5, 10$.

and thus we would not be able to apply existence theorems or numerical integration. Hence, we use the logistic function as a smooth approximation of H:

$$H_k(C) = \frac{1}{1 + (\frac{1}{H_0} - 1) \cdot e^{k(C - C_{max})}}$$

where $H_k(C_{max}) = H_0 > 0$. Note that:

$$\begin{aligned} \int_{-\infty}^{+\infty} (H(x) - H_k(x))^2 dx &= \int_{-\infty}^{C_{max}} (1 - H_k(x))^2 dx + \int_{C_{max}}^{+\infty} (H_k(x))^2 dx \\ &= 2 \cdot \int_{C_{max}}^{+\infty} (H_k(x))^2 dx \leq \frac{1}{k}. \end{aligned}$$

Hence $H_k \to H$ as $k \to +\infty$ in the L^2 norm. This convergence is shown in Figure 4.1.

Therefore we obtain the following *logistic finite-queue* model:

$$\begin{cases} C'(t) &= \tilde{f}(C(t),t) - \left[\mu + e^{-\alpha C(t)}\left(\min\{\mu, \tilde{f}(C(t),t)\} - \mu\right)\right] \text{ for } t > t_0, \\ C(t_0) &= C_0. \end{cases} \tag{4.1}$$

where

$$\tilde{f}(C(t),t) = H_k(C(t)) \cdot f(t).$$

Theorem 4.2.1. *Let $C(t_0) = C_0 \geq 0$ be an initial condition, $f \geq 0$ a continuous and positive inflow function, $\mu > 0$ a maximum outflow rate and $C_{max} > 0$ a maximum queue capacity. Then we have that the system (4.1) satisfies the following properties:*

1. *There exists a unique an positive solution defined for all times.*
2. *If the inflow is strictly smaller than the maximum outflow rate then the queue gets empty exponentially fast.*
3. *FIFO: the exit time of an entity that has arrived in the instant t is smaller than other that arrives at $s > t$.*
4. *If $C_0 < C_{max}$ and the inflow satisfies that*

$$f(t) \leq M_f \text{ for all } t \geq t_0 \text{ for some } M_f > 0, \tag{4.2}$$

then there exists a $H_0 > 0$ and a $k > 0$ such that H_k approximates H as precisely as desired (in the L^2 sense) and moreover

$$C(t) \leq C_{max} \text{ for all } t \geq t_0.$$

Proof. Properties 1,2 and 3 follow because the right hand side of (4.1) is *locally Lipschitz continuous* [15] so we can apply again existence and uniqueness, and afterwards repeat analogous arguments like the ones given in the previous chapter. Let's prove property number 4. Assume there is a $t' > t_0$ such that $C(t') > C_{max}$. Since $C(t_0) < C_{max}$, there must exist a t_* such that $C(t_*) = C_{max}$. We shall assume that in a sufficiently small interval I around t_* the function $C(t)$ is strictly increasing, so that $C'(t) > 0$ for $t \in I$. Analogously as in the proof of Proposition 3.2.1 we must have that $\tilde{f} > \mu$ in I. Hence

$$\begin{aligned} C'(t_*) &= \tilde{f}(C(t_*),t_*) - \mu = H_0 \cdot f(t_*) - \mu \\ &\leq H_0 \cdot M_f - \mu = 0 \end{aligned}$$

where we have taken $H_0 = \frac{\mu}{M_f}$. Whence we get a contradiction, so the queue remains bounded by C_{max} for all times. □

4.3 Multiple servers

Assume we have k servers of speed μ_0. When an entity arrives to the system it goes to the first server that finds empty and else goes to the queue. The key modification of (3.3) now, is to make $\mu = \mu(C(t))$ dependent of the queue size. We take:

$$\mu(C(t)) = \begin{cases} \mu_0 k & \text{if } C(t) \geq k-1, \\ \mu_0 \left(1 + C(t)\right) & \text{if } C(t) \leq k-1. \end{cases} \tag{4.3}$$

The idea behind this equation is the following: if there are no entities in queue and one arrives it gets served at speed μ_0 as usual. On the other hand, if there are, say, 5 entities in queue and 1 more arrives, we should have 5+1 servers functioning in order to serve all 6 entities. Hence the total speed of the system would be $6\mu_0$.

4.4 Separation of flows

Assume f_1 and f_2 are two inflows that join in a system with a server and a queue as in Figure 4.2. They are processed in that system but afterwards they follow different paths. Denote by g_1 and g_2 the corresponding outflows.

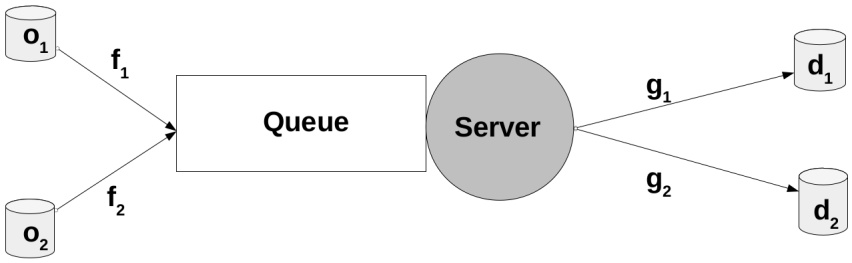

Figure 4.2: Two flows join in a server with a queue and then each of them follow a different path.

We can obtain them with a very simple argument. If we denote by $f = f_1 + f_2$ and process this inflow, we get an aggregated outflow g. Then, for each t we should have that:

$$\frac{f_1(t)}{f(t)} = \frac{g_1(t)}{g(t)},$$

hence

$$g_1(t) = \tfrac{g(t)}{f(t)} \cdot f_1(t).$$

Likewise,

$$g_2(t) = \tfrac{g(t)}{f(t)} \cdot f_2(t).$$

Note that this way we have that $g_1(t) + g_2(t) = g(t)$ as expected. A similar argument can be given in the case we have more than two flows.

4.5 Priority queues

Assume again f_1 and f_2 are two inflows that join in a system with a server of fixed speed μ. The only difference is that this time we assume that the flow f_1 has a priority over f_2, i.e. entities coming from f_1 will be served first than entities from f_2. We will model separately the queues formed by each flow:

$$\begin{cases} C_1'(t) &= f_1(t) - \left[\mu_1 + e^{-\alpha_1 C_1(t)} \left(\min\{\mu_1, f_1(t)\} - \mu_1\right)\right], \\ C_2'(t) &= f_2(t) - \left[\mu_2 + e^{-\alpha_2 C_2(t)} \left(\min\{\mu_2, f_2(t)\} - \mu_2\right)\right], \end{cases} \tag{4.4}$$

where we impose that $\mu_1 + \mu_2 = \mu$. The key idea now is to take:

$$\mu_2(C_1(t)) = \frac{f_2(t)}{f(t)} \mu e^{-\beta C_1(t)}$$

where $f = f_1 + f_2$. Also we take $\mu_1(C_1(t)) = \mu - \mu_2(C_1(t))$. Observe that:

1. If $C_1 = 0$ then $\mu_1 = \frac{f_1}{f}\mu$ and $\mu_2 = \frac{f_2}{f}\mu$ so each speed is proportional to its inflow weight.
2. If $C_1 \to +\infty$ then $\mu_1 \to \mu$ and $\mu_2 \to 0$ so just the priority one entities are served and the others are not.

4.6 Summary

In this chapter we have presented variations of the basic logistic queue model that allow us to address more general scenarios present in telecommunication networks. This is one of the main advantages of our model over others. We have shown how we can modify slightly the main equation to include important cases such as: finite queues and priority queues. This will be useful in chapter 6 when we present an application of the model in a network.

Chapter 5

Validation and Results

In this chapter we validate the logistic queue model comparing its performance with a discrete event simulator.

5.1 Notations

In this section we recall all our notation and its meaning. They will be used throughly in the following sections.

- *Maximum outflow rate*: μ. Its units are entities/time. It will be assumed to be constant, for example, $\mu = 1$ Gb/s. It is given, so it is an input.
- *Queue size at t*: $C(t)$. Its units are entities. It varies over time. Our goal is to predict this quantity.
- *Inflow to the system at t*: $f(t)$. Its units are entities/time. It varies over time. It is given, so it is an input.
- *Outflow of the system at t*: $g(t)$. Its units are entities/time. It varies over time. Our goal is to predict this quantity.
- *Mean inflow*: λ. Its units are entities/time. It is the mean of the inflow f.
- *Mean intensity or occupancy*: ρ. It has no units. It is defined as $\frac{\lambda}{\mu}$. This quantity gives us an idea of the average use of the server.
- *Aggregation time*: dt. Its units are time. When a flow is given in discrete form, we will assume it is given each dt seconds. Usually $dt = 60$.
- *Logistic model parameter*: α. It is defined as $\frac{\rho}{\mu}$.

5.2 Error measures

In the following sections we will compare the queue size given by the logistic queue model, which we denote $C_{log}(t)$, with the queue size given by a discrete packet simulator, which we denote $C_{disc}(t)$. In order to compare them we need some error measures. Most of the time those quantities are given at discrete instants of time. So in the end we will only have two vectors $\mathbf{C_{disc}}, \mathbf{C_{log}}$ evaluated at a finite number of times n. Since the logistic model will not react to small fluctuations in the queue, we will be mostly interested in checking if the model captures big queues. These are the important ones because they affect the outflow the most. We define:

1 *Error relative to the maximum*:

$$\frac{\|\mathbf{C_{disc}} - \mathbf{C_{log}}\|}{\|\mathbf{1} \cdot \max(\mathbf{C_{disc}})\|}$$

where $\mathbf{1}$ is vector of ones of length n.

2 *Maximum occupancy error*:

$$\frac{|\max(\mathbf{C_{disc}}) - \max(\mathbf{C_{log}})|}{|\max(\mathbf{C_{disc}})|}.$$

Likewise, we will want to compare the outflow given by the logistic queue model, which we denote $g_{log}(t)$, with the outflow given by the discrete Simulink simulator, which we denote $g_{disc}(t)$. Again, in order to compare them we need some error measures. Most of the time those quantities are given at discrete instants of time. So in the end we will only have two vectors $\mathbf{g_{disc}} = (g^1_{disc}, \ldots, g^n_{disc})$ and $\mathbf{g_{log}} = (g^1_{log}, \ldots, g^n_{log})$ evaluated at a finite number of times n. We define:

3 *Mean relative outflow error*:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|g^i_{disc} - g^i_{log}|}{|g^i_{disc}|}$$

4 *Global relative error*:

$$\frac{\|\mathbf{g_{disc}} - \mathbf{g_{log}}\|}{\|\mathbf{g_{disc}}\|}$$

5.3 User service characterization

In order to test the logistic queue model, we will model and simulate realistic traffic flows. In the following we will focus in video flows due to the easiness of characterization, but the extension to other kind of services is straightforward. We will model the inflow created by a video user in a discrete way, i.e. packet by packet.

For characterizing video streaming, an on-demand video file was served from a set of HTTP servers to a single end user based on the MPEG-DASH v1.4 standard. On the server-side, two virtual machines each running an Apache HTTP server instance were responsible for serving the audio and the video components, respectively. The video was served at HD 720p and its duration was 10 minutes.

Now, we will make some statistical assumptions about the packets generated while using a video service:

1. All packets are assumed to be of the same constant size.
2. Packets are assumed to come in *bursts* of variable size. We will assume that the size of each burst follows a *Normal distribution*.
3. Bursts are assumed to be separated by variable intervals of time. We will assume that the size of an interval of time separating two bursts –*Interburst time*– follows an *Exponential distribution*.
4. Finally, packets are assumed to be separated by variable intervals of time inside a burst. We will assume that the size of an interval of time separating two packets inside a burst –*Interpacket time*– also follows an *Exponential distribution*.

Once the real video flows were generated, the parameters of each distribution were estimated using standard statistical methods. We obtained the results of Table 5.1.

Table 5.1: Estimation of parameters for a video user.

Packet size (Bytes)	1464
Burst size (mean)	1714
Burst size (variance)	278
Interburst time (seconds)	5.56
Interpacket time (seconds)	0.00345

Next we want to model the use of video service. Since we are interested in simulating more than a few hours, we cannot assume that each video user is watching video continuously without stopping.

Hence we assume that *uses of the service* are separated by an interval of time –*Interuse time*– following an *Exponential distribution* with mean 45 minutes. So in mean, a typical user watches some videos each forty five minutes. Finally, we know from experience that not all videos are of the same length, and even if they were, it is possible and usual to watch more than one. Therefore we also assume a probability distribution on *video consumptions* listed in Table 5.2.

Table 5.2: Length of video consumptions and their probabilities.

Video use (minutes)	*Probability*
5	0.4
15	0.3
30	0.25
120	0.05

Using all previous assumption we can simulate all packets generated by a video user. Its easier to picture all the previous concepts with a figure. In Figure 5.1 we show a simulated packet flow using the previously estimated parameters.

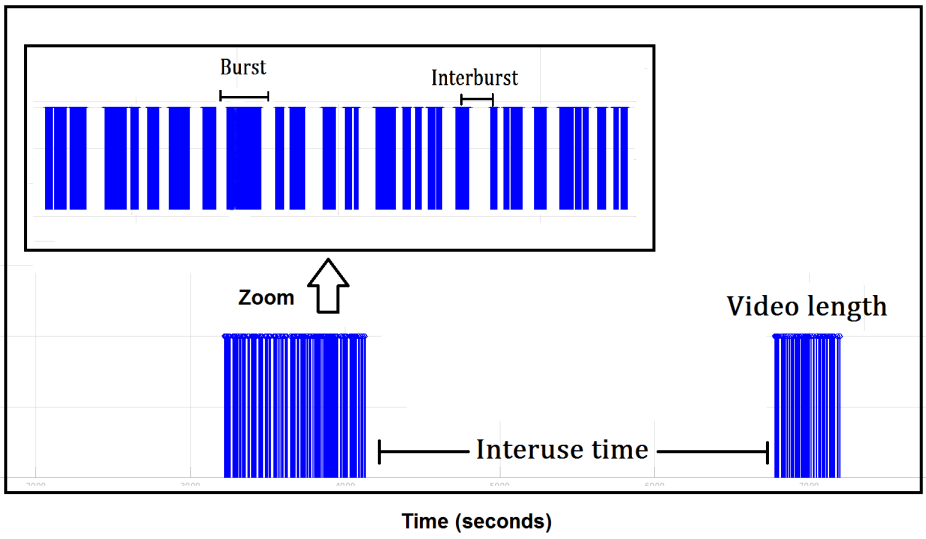

Figure 5.1: Simulation of packets generated by a video user during two hours. Each blue line represents a packet.

Now we want to compute the flow f_v it generates. In order to do this, we simply count how many packets have occurred each $dt = 60$ seconds and then add them. In more mathematical terms, in a point of the form $t_{i+1} = (i+1) \cdot dt$ we put

$$f_v(t_{i+1}) \approx \frac{1}{dt} \sum (\text{packets occurring in } [i \cdot dt, (i+1) \cdot dt]) .$$

In Figure 5.2 we plot the flow generated by a typical video user. As it is expected, the service is not being used constantly, but rather sparely as wanted.

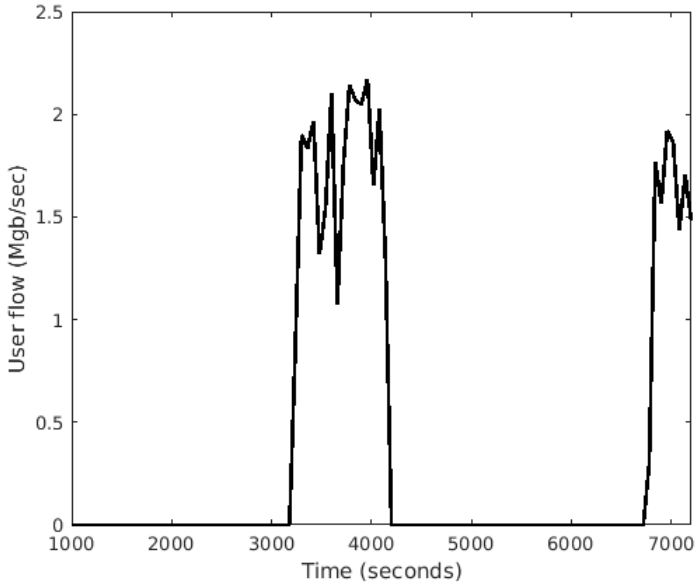

Figure 5.2: Flow f_v generated by one video user during two hours.

5.4 Implementation

As already said, in the following we will compare the queue size given by the logistic queue model with the queue size given by a discrete packet simulator. We implemented both models using standard software:

1. The logistic queue model was implemented in MATLAB. The only thing that has to be done is to solve the ordinary differential equation (3.3). In Matlab there are several libraries for solving differential equations, being the most usual ones *ode45* and *ode113*. Both functions give indistinguishable results.

2. The discrete packets simulator was implemented in SIMULINK. Simulink is a block diagram environment for multidomain simulation and Model-Based Design. It supports simulation, automatic code generation, and continuous test and verification of embedded systems. We can see a screenshot of part of the implemented system in Figure 5.3.

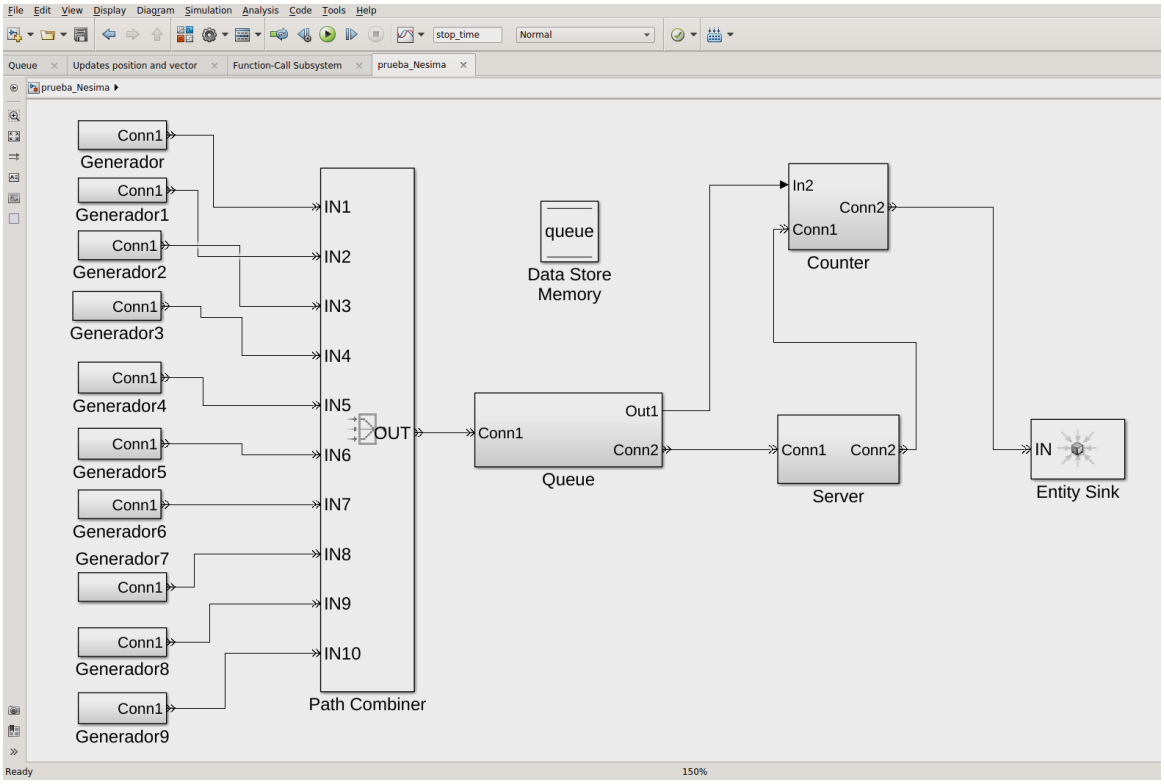

Figure 5.3: Implementation of a queue with a server in Simulink.

5.5 Illustrative example

In order to have an intuitive understanding of the model and all the things involved with it, we will first give an illustrative example of its performance. Later we will make a more exhaustive performance analysis. We assume the following particular scenario:

1. We aggregate 10 video users in a server of maximum velocity $\mu = 11.33$ Mb/s during 2 days.

2. The queue is assumed to be large enough so that we do not lose any packets.

For generating the video flows we use the method described in the previous section. The goal is to predict the queue formed under this scenario and the outflow we get.

Inflow

Adding the 10 video flows we get the inflow of Figure 5.4. In other words we put

$$f(t) = \sum_{v=1}^{10} f_v(t).$$

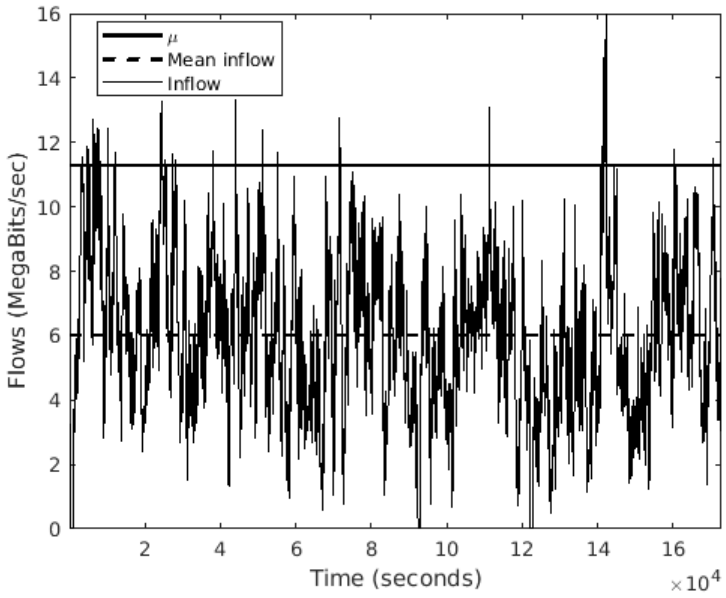

Figure 5.4: Inflow to the system. Its standard deviation is 2.53 Mb/s.

The mean of the inflow is $\lambda = 6$ Mb/s, hence the average intensity is $\rho = 53\%$. Therefore we take $\alpha = \frac{\rho}{\mu} = 0.05$. As we see in the picture, during some periods of time, the inflow is bigger than the maximum outflow μ, therefore in those moments we expect the queue size to increase.

Comparison of queues

We feed the inflow f to both the Simulink discrete model and to the logistic queue model and compare the queue results we get in both cases. Note that f is given to the discrete simulator in discrete form, i.e. packet by packet as it was generated, whereas to the logistic model the inflow is given in a continuous fashion, i.e. aggregated each 60 seconds and then linearly interpolated when needed. Just looking at Figures 5.5 and 5.6 of the queues formed is impossible to tell the difference, hence we need some metrics to compare them. We will use the two measures introduced before:

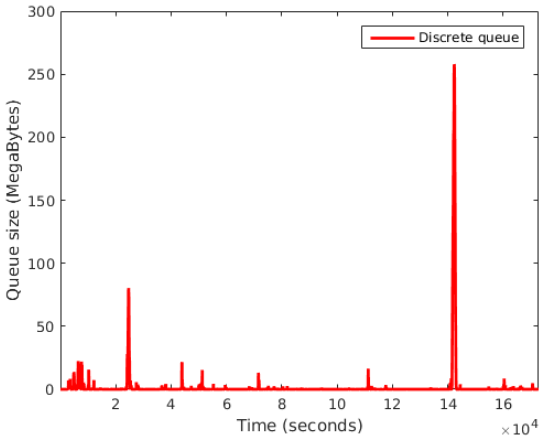

Figure 5.5: Queue of the discrete simulator.

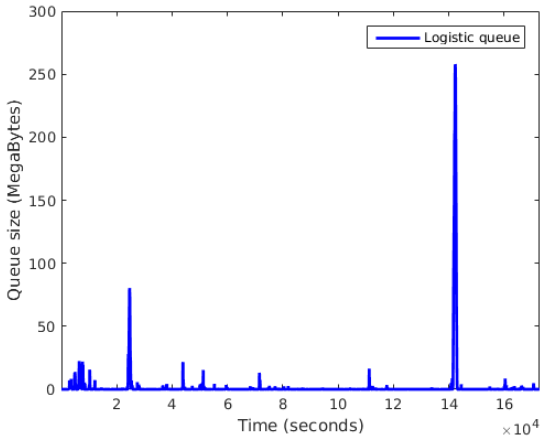

Figure 5.6: Queue of the logistic model.

- *Error relative to the maximum*:

$$\frac{\|\mathbf{C_{disc}} - \mathbf{C_{log}}\|}{\|\mathbf{1} \cdot \max(\mathbf{C_{disc}})\|} = 0.63\%$$

- *Maximum occupancy error*:

$$\frac{|\max(\mathbf{C_{disc}}) - \max(\mathbf{C_{log}})|}{|\max(\mathbf{C_{disc}})|} = 2.08\%$$

It is worth to examine both queues superposed in a neighborhood of the maximum occupancy. This is shown in Figure 5.7.

Comparison of outflows

We can also compare both outflows, the one given by the discrete simulator and the other computed from the logistic queue. As expected both flows do not exceed the maximum capacity μ. Just looking at Figures 5.8 and 5.9 is again impossible to say anything. We again use the two measures already introduced:

- *Mean relative outflow error*:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|g^i_{disc} - g^i_{log}|}{|g^i_{disc}|} = 0.67\%$$

- *Global relative error*:

$$\frac{\|\mathbf{g_{disc}} - \mathbf{g_{log}}\|}{\|\mathbf{g_{disc}}\|} = 1.58\%$$

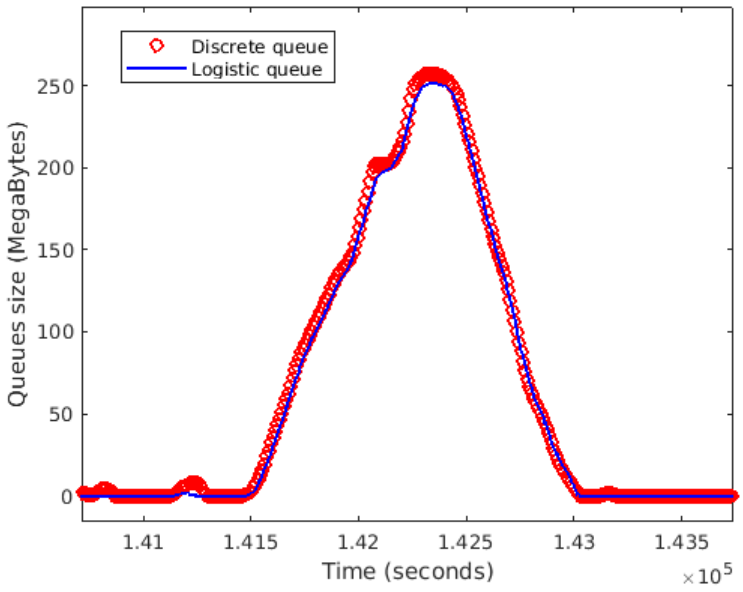

Figure 5.7: Superposition of logistic and discrete queues in a neighborhood of the maximum queue occupancy.

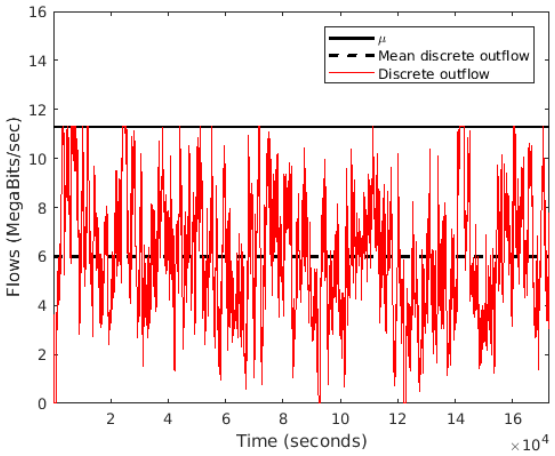

Figure 5.8: Outflow of discrete simulator.

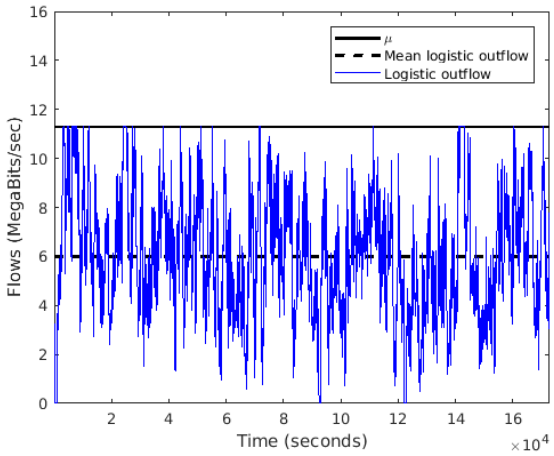

Figure 5.9: Outflow of logistic model.

Comparison with no queue model

In order to have a reference to compare with, we can compute the previous error measures with the inflow to the system. This would be the simplest model in which we assume that the outflow equals the inflow and the queue has no effect whatsoever. In this example the

mean relative error is:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|g^i_{disc}-f^i|}{|g^i_{disc}|}=0.75\%,$$

while the global relative error is:

$$\frac{\|\mathbf{g_{disc}}-\mathbf{f}\|}{\|\mathbf{g_{disc}}\|}=5.27\%.$$

As expected mean-wise the relative errors are very similar. This is due to the fact that most of the time the inflow is smaller than μ and hence the outflow g is equal to f. On the other hand the global relative error is sensibly bigger for the no queue model since this measure penalizes more point-wise errors.

Discussion of example

As already stated the goal of this example was to introduce in an intuitive fashion all concepts involved with the model in a practical setting. The results are very promising in the sense that our model captures very well the queue generated and hence also the outflow. At the end of the day, one may wonder why it is worth introducing a continuous model (the logistic queue model) if we already can make discrete simulations and get quite accurate results. The key is of course **scalability**. In this illustrative example the simulation time of the discrete simulator was around 60021 seconds, which is about 16 hours. On the other hand to the logistic queue model the whole process only took about 29 seconds. So the continuous model is around 2000 times faster! This is expectable since the discrete simulator had to process around $2.45 \cdot 10^7$ entities. In the next section we will make a more comprehensive analysis of the performance of the model.

5.6 Performance Analysis

In this subsection we analyze how the error measures introduced in the previous examples evolve as the intensity of the inflow is varied. Again we assume the same scenario as in the example where we aggregate 10 video users in a server of maximum velocity $\mu = 11.33$ Mb/s during 2 days each 60 seconds.

We vary inflow intensities ρ from approximately 45% to 85%. In order to do so we simply reduce the *Interuse time* introduced before when modeling a video user. Recall that this parameter represented the amount of time between two uses of video service. Hence reducing it, we only make our 10 users watch videos more often. In total 21 simulations were launched both in the discrete event simulator implemented in Simulink and using the logistic queue model implemented in Matlab. Of the 21 simulations 3 of them had not finished after more than a month in the discrete simulator, and therefore they were discarded. Note that in

each simulation we are generating around 25 million packets, so in total in 21 simulations we generated about 500 million packets, which is roughly about 6000 GigaBytes of data.

Comparison of queues

We first compare the maximum queue size given by both models. In Figure 5.10 we can see a plot of both curves for different intensities.

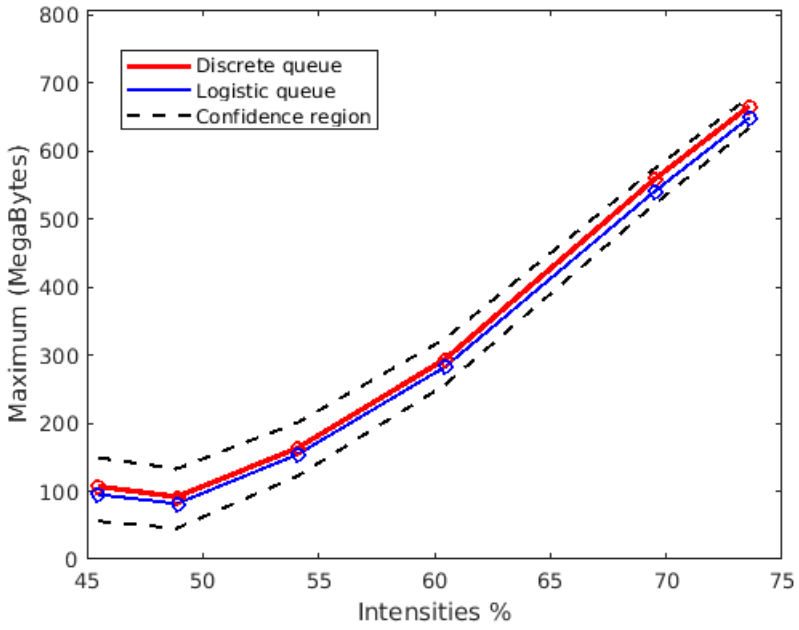

Figure 5.10: Comparison of maximum queue size for both models varying the inflow intensity.

As we see both curves are quite close. In dashed lines we have plotted the error bound (3.16).

It is also interesting to see how the maximum occupancy error evolves as the intensity increases. We see in Figure 5.11 how the error decreases as the intensity increases. In Figure 5.12 we plot the error relative to the maximum introduced earlier. Again the error reduces as the intensity increases. This is again in accordance with (3.16).

Comparison of outflows

Next we compare the outflow given by the logistic queue model with the one given by the discrete simulator. In order to have a reference to compare with, we also plot the error of

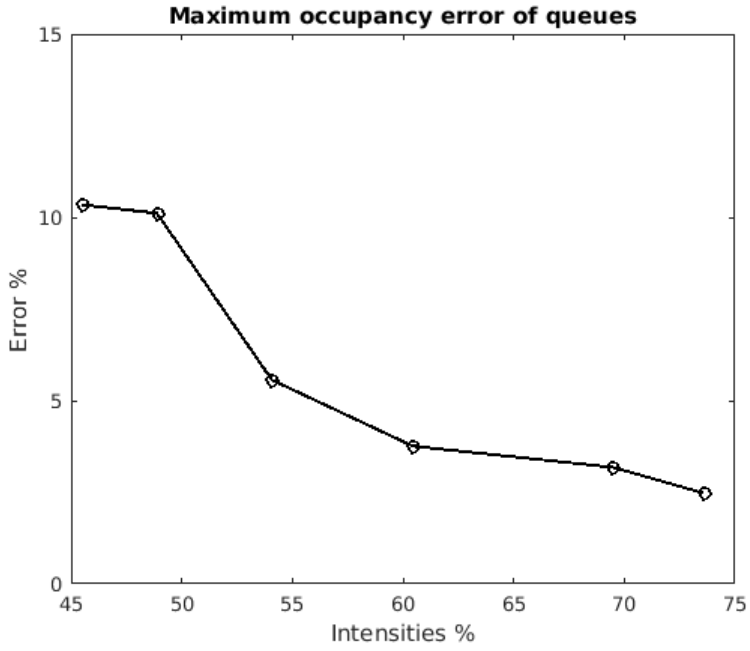

Figure 5.11: Maximum occupancy error of the logistic model varying the inflow intensity.

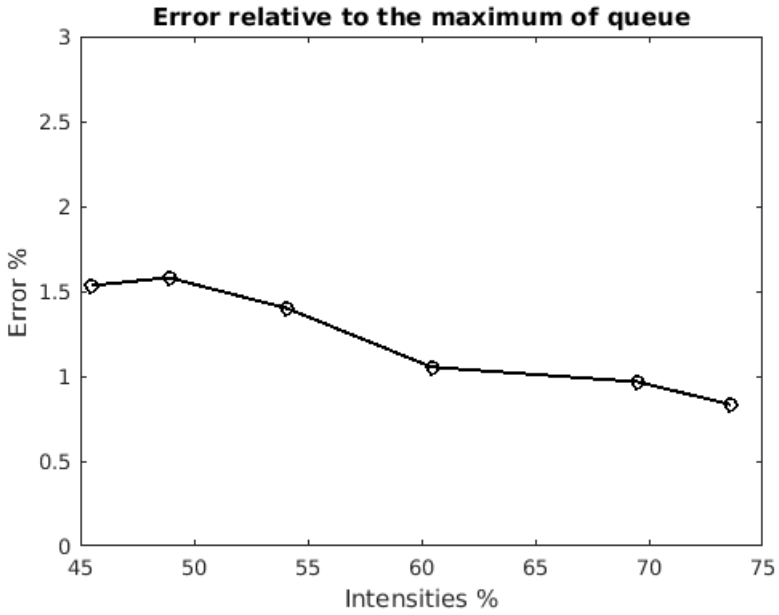

Figure 5.12: Error relative to the maximum of the logistic model varying the inflow intensity.

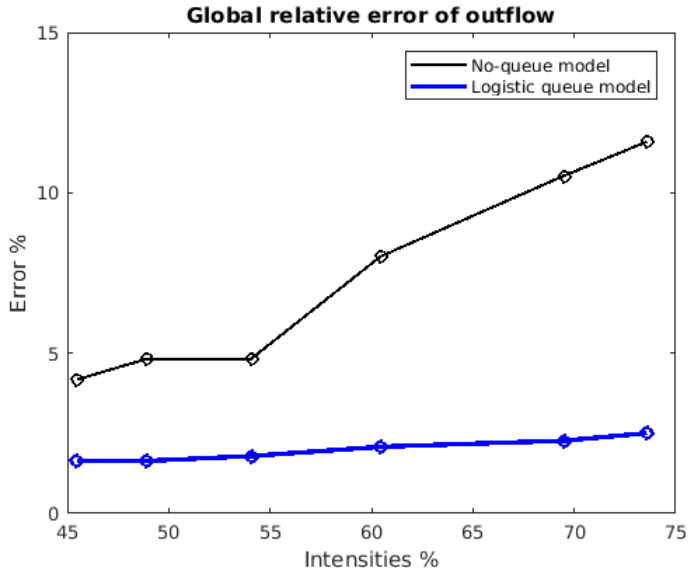

Figure 5.13: Global relative error of the outflows given by logistic model and the no-queue model varying the inflow intensity.

the **no-queue model** as discussed before. Recall that this just assumes that the queue has no effect whatsoever so the outflows just equals the inflow. We begin by plotting the global relative error in Figure 5.13.

Note how the error of the no-queue model increases as the intensity increases. This is to be expected because increasing the intensity increases the queue size and therefore makes the outflow differ more from the inflow. On the other hand the error of the logistic queue model seems quite stable, its mean is around 2%.

Finally we compute the mean relative error. In Figure 5.14 we observe that until $\rho = 55\%$ there is no big difference between both models, both errors are quite low. This is again logical since we know that the outflow of the system is almost identical to the inflow when there are not a big queues.

Scalability: simulation times

Now we want to compare the simulation time of both models. Since the mean simulation time of the logistic model is 26.3 seconds but the mean of the discrete simulator is 7.6 days we apply the function $\log_{10}$.

As we see in Figure 5.15, the logistic queue model is about 5 orders of magnitude faster than the discrete simulator when the intensity of the inflow is about 60%. This is a huge difference, the continuous model is $100,000$ times faster than the discrete one.

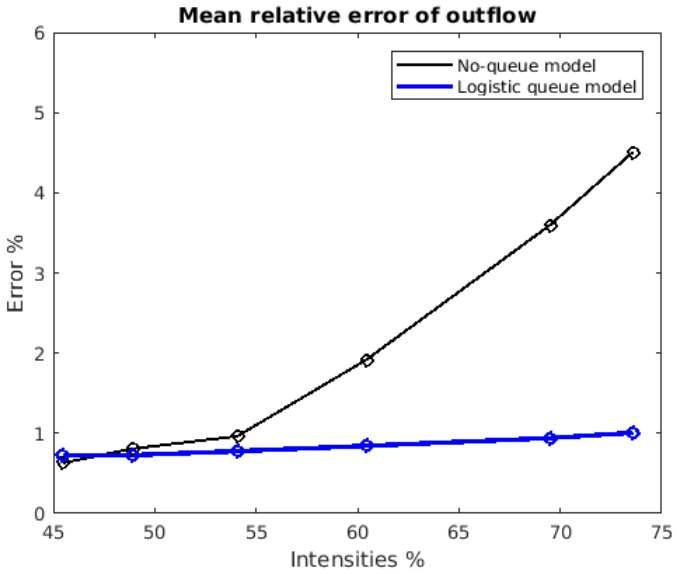

Figure 5.14: Mean relative error of the outflows given by logistic model and the no-queue model varying the inflow intensity.

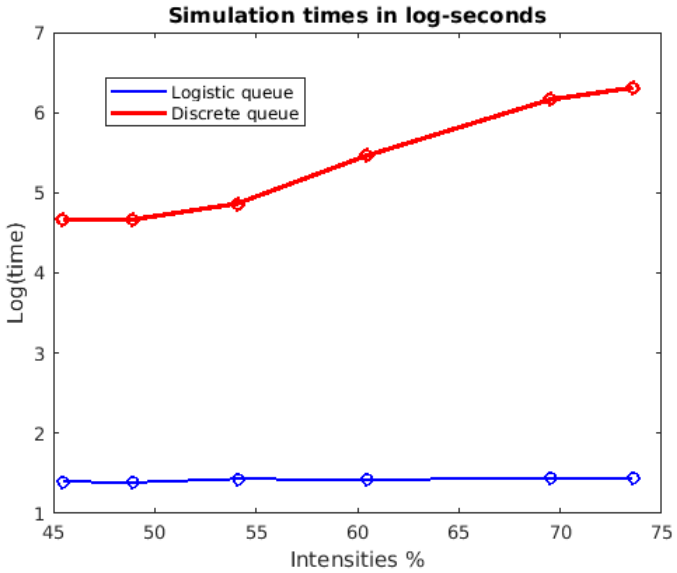

Figure 5.15: Simulation times of logistic and discrete queue model varying the inflow intensity.

5.7 Summary

In this chapter we have validated the logistic queue model comparing its performance with a discrete event simulator. We have shown that:

- Our model is as precise as the discrete one. We compared queue sizes and outflow predictions and the results show that differences are minimal. For example, the global relative error of the outflow given by the logistic model is only about 2%.
- Our model is several orders of magnitude faster than the discrete one. For example, when the mean intensity of the inflow is 60% the logistic model is about 100,000 times faster than the discrete one.

In the next chapter we will give an application of the logistic queue model in a scenario where a discrete simulation would be unfeasible. We will be dealing with flows of the order of Gb/s.

Chapter 6

Case study example

In this chapter we present a case study using the logistic queue model in a telecommunication network. Basically we study how the latency of the network is affected when we introduce an external priority flow into play. We will be dealing with flows of the order of Gb/s, this is where our model becomes useful, since running a discrete simulation for these orders of magnitude would be impossible.

6.1 Scenario

For the rest of this chapter we will assume the scenario shown in Figure 6.1:

1. A set of flows $f_1, \ldots, f_n$ are generated from secondary packet nodes o_i of an optical network. We assume we have n nodes o_i.

2. A proportion p_{ij} of each flow f_i has to be sent from o_i to a specific packet node d_j at the other side of the network. We assume we have m nodes d_j.

3. This is done through a big connection from an origin node O to a destination node D. This link has a server of speed μ with a queue attached to it of maximum capacity C_{max}.

4. Each flow f_i travels through a link between the node o_i and the node O. That given connection has a a server of velocity μ_i and a queue. These queues are assumed to be large enough so that we do not lose any packets.

5. Finally we assume that links between the node D and the destination nodes d_j have a server of velocity ξ_j and a queue. Likewise, these queues are assumed to be large enough so that we do not lose any packets.

In terms of entities (or packets) our network is equivalent to the one depicted in Figure 6.2.

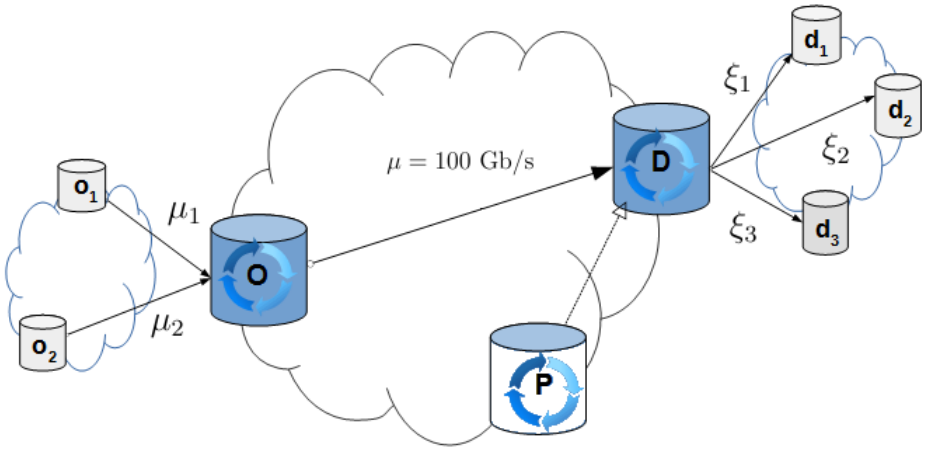

Figure 6.1: Network for the scenario in flow form. The goal is to send packets from region o to region d of the network. In the picture $n = 2$ and $m = 3$.

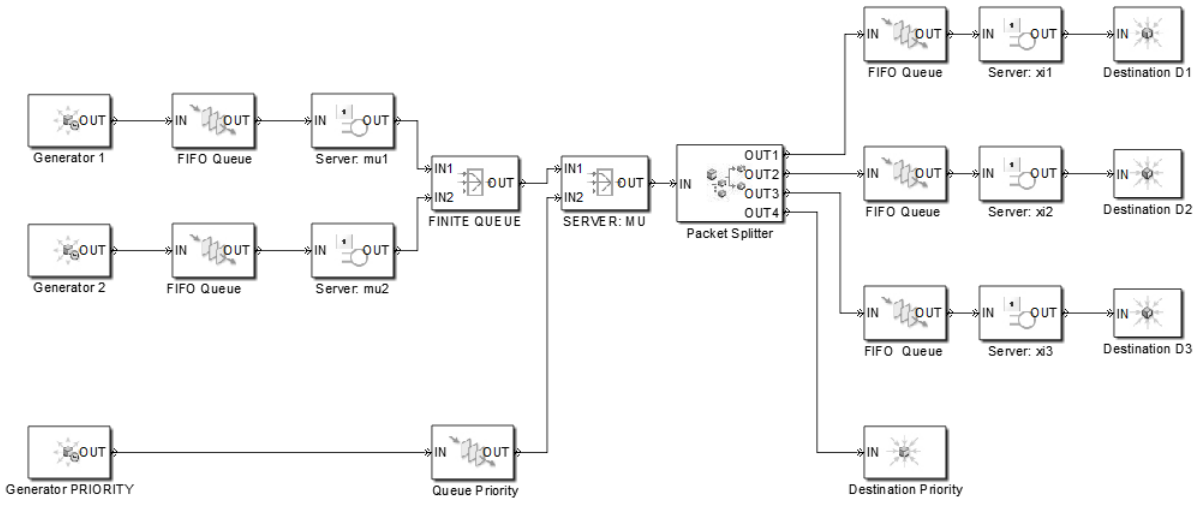

Figure 6.2: Network for the scenario in packet form. The goal is to send packets from region o to region d of the network. In the picture $n = 2$ and $m = 3$.

We will be mainly interested in computing the **latency** $L_{ij}(t)$ between node o_i and node d_j at instant t. Physically this represent the amount of time it takes to a packet to travel

from o_i to d_j if it departures at t. In our system we have that:

$$L_{ij}(t) = \frac{C_i^o(t) + s}{\mu_i} + \frac{C(t_o) + s}{\mu} + \frac{C_j^d(t_d) + s}{\xi_j},$$

where

- s is the size of each packet.
- $C_i^o(t)$ is the queue formed between node o_i and O at t.
- $C(t_o)$ is the queue formed between node O and D at t_o.
- t_o is the instant at which the packet arrives to O: $t_o = t + \frac{C_i^o(t)+s}{\mu_i}$.
- $C_j^d(t_d)$ is the queue formed between node D and d_j at t_d.
- t_d is the instant at which the packet arrives to D: $t_d = t_o + \frac{C(t_o)+s}{\mu_i}$.

It is easy to see that the previous formula can be generalized for more general networks. Moreover we define the **expected latency** of going from region o of the network to region d as:

$$L_{od}(t) = \frac{1}{n \cdot m} \sum_{i,j} L_{ij}(t).$$

Finally we define the **maximum expected latency** as

$$L_{max} = \max_t L_{od}(t).$$

We will be interested in studying how this quantity changes when we modify the scenario just described above.

6.2 Applying the logistic queue model

Applying the model in the scenario just described is quite straightforward. First we process each flow f_i using the logistic queue model (3.3) to obtain flows g_i. These are the outflows of each o_i and the inflows to O. Doing this we obtain the functions $C_i^o(t)$. Then we put $g := \sum_{i=1}^n g_i$ and process this flow using the logistic finite queue mode (4.1) to obtain a new flow h. This is the outflow of O and the inflow to D. Doing this we obtain the function $C(t)$.

Now, we must take into account the origin o_i of each flow and its destination d_j, hence by (4.4):

$$h_j(t) = \sum_{i=1}^{n} p_{ij} \cdot \left(\frac{g_i(t)}{g(t)} \cdot h(t) \right).$$

This is the outflow of D and the inflow to d_j. Finally we process each h_j using again the the logistic queue model (3.3). Doing this we obtain the functions $C_j^d(t)$.

6.3 Empirical results

- For generating the flows f_i we use the method described in section 5.3 when we modeled a video user. In total $n = 4$ flows were generated during 1 day.

- Each flow f_i consists of 10,000 video users, the mean inflow rate of each flow is about 12.5 Gb/s. We take all $\mu_i = 25$ Gb/s.

- For the big link OD we take $\mu = 100$ Gb/s and $C_{max} = 25$ GigaBytes.

- Since the mean of each flow f_i is about 12.5 Gb/s and $n = 4$, we expect the use of the OD link to be around 50%.

- We take $m = 5$ nodes d_j. Each link has a speed of $\xi_j = 20$ Gb/s.

- Finally, the matrix p_{ij} is chosen to be:

$$p = \begin{bmatrix} 0.1293 & 0.3124 & 0.0548 & 0.2534 & 0.2501 \\ 0.1600 & 0.1681 & 0.0497 & 0.2203 & 0.4019 \\ 0.3029 & 0.0009 & 0.1687 & 0.2224 & 0.3051 \\ 0.0042 & 0.3710 & 0.2344 & 0.0250 & 0.3655 \end{bmatrix},$$

so, for instance, 12.93% of the packets generated from node o_1 go to node d_1.

Then, in this setting we get the expected latency of Figure 6.3. We see that for a packet that departs from region o of the network we expect that it takes at most 0.2 seconds to get to its destination in region d. The whole simulation took only about 2 minutes to be completed.

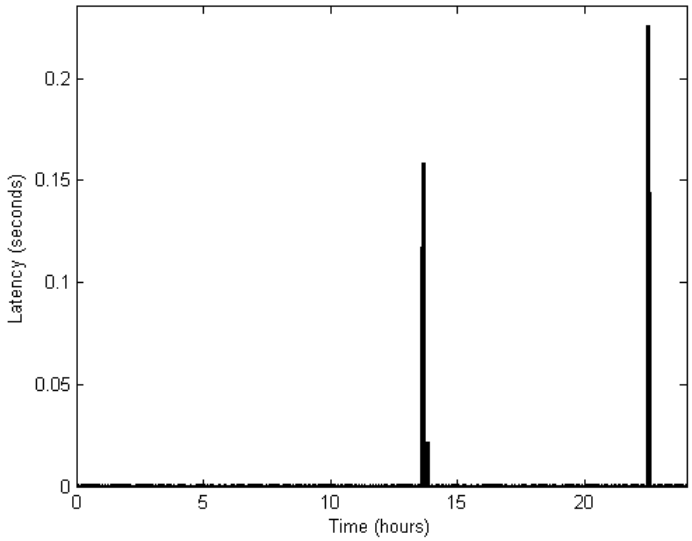

Figure 6.3: Expected latency in seconds of going from region o to region d.

Change in the scenario

Now, it may be the case that another flow of packets needs to be sent to D using the same link OD that the node O was using as in Figure 6.2. This is represented in Figure 6.1 by the node P. We will assume that the flow going from P to D has priority over the flow going from O to D. We implement this using model (4.4): we inject an additional priority flow into the system. We are interested in seeing how this will affect the maximum expected latency of going from region o to region d. The results, varying the intensity of the priority flow are shown in Figure 6.4.

As expected the latency gets bigger as the priority flow gets bigger. Nevertheless it is interesting to observe that for priority flows smaller than 18 Gb/s we get a tolerable maximum expected latency of about 0.5 seconds. Afterwards the increase is exponential and we get unacceptable latencies.

6.4 Summary

In this chapter we have seen how we could apply the main ideas developed along the project to a telecommunication network. We have only scratched the surface, showing only the potential of the model. Along the way we have seen how we can calculate a quantity of interest in the network: the latency. This in contrast to other network models that do not

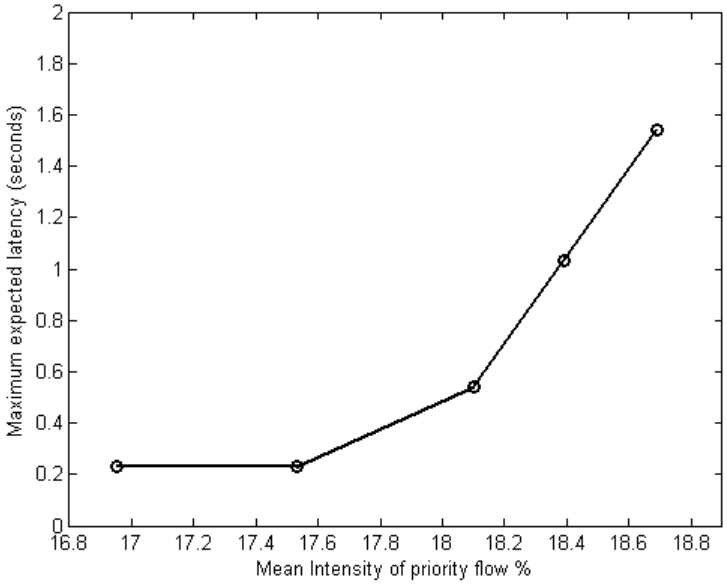

Figure 6.4: Maximum expected latency in seconds of going from region o to region d varying the intensity of an extra priority flow in the simulation.

modelize queues of servers as stated in the introduction.

Chapter 7

Concluding Remarks

7.1 Contributions archieved

In this work we have presented a novel continuous queuing model called the *logistic queue model*. We have proven mathematically that our proposed model has many desirable theoretical properties. Moreover, we validated the model comparing it with a discrete event simulator. We showed that in terms of queue size and outflow prediction our model is as precise as a discrete one with the advantage of speed in simulations. We compared simulation times and concluded that the logistic queue model is several orders of magnitude faster than the discrete one. Finally in contrast with the point-queue model our model allow us to easily explore multiple extensions to more general scenarios such as: finite queues, multiple servers, priority queues, etc. This in turn allowed us to show an application of all the tools developed.

7.2 Personal evaluation

Throughout this project I have applied many skills developed in the master program (MAMME). Specially useful have been the courses regarding modelling with differential equations and others related to the numerical solution of them. Moreover, elective courses taken from the statistics master (MESIO) have proven to be of conceptual help in the development of the thesis. Also, I have had to learn from scratch other important topics like discrete event simulators and the functioning of dynamic optical networks. This project enabled me to participate in the research Grup de Comunicacions Òptiques (GCO) was conducting. I have had the opportunity of learning how to read and compare research papers, how to study some topics that I had not covered during my university courses, how to improve my programming skills (mostly in Simulink), how to modelize using mathematical tools, as well as the capacity to solve relevant real world problems.

7.3 Future work

This work will be partially included as UPC contribution to the H2020 European METRO-HAUL project (G.A. nº 761727). Specifically, a traffic generation tool will be defined and implemented for creating realistic traffic traces exchanged in/between optical metro networks (metro flows). The tool (to be developed in Python) will basically consists in: i) a graphical user interface where different configuration parameters can be tuned, e.g. characteristics and magnitude of individual user/service flows, periodic pattern, duration, etc; ii) a traffic generation engine that will receive input parameters to output metro flow traffic traces; and iii) a visualization module to easily check generated traffic flows, as well as different export formats to facilitate the use of data for multiple use cases including network optimization and analytics. The main contributions of this master thesis will be used to build a fast and accurate traffic generation engine based on the logistic queue model.

Bibliography

[1] CISCO Global Cloud Index (GCI), 2015.

[2] CISCO Visual Networking Index (VNI), 2016.

[3] L. M. Contreras, V. López, O. González, A. Tovar, F. Muñoz, A. Azañón, J.P. Fernández-Palacios, and J. Folgueira: *Towards cloud-ready transport networks.* IEEE Communications Magazine, vol. 50, pp. 48-55, 2012.

[4] D. King and A. Farrel: *A PCE-based architecture for application-based network operations.* CIETF RFC7491, 2015.

[5] V. Lopez and L. Velasco: *Elastic Optical Networks: Architectures, Technologies, and Control.* Ed. Springer, 2016.

[6] L. Velasco, A. Castro, M. Ruiz, and G. Junyent: *Solving Routing and Spectrum Allocation Related Optimization Problems: from Off-Line to In-Operation Flexgrid Network Planning.* IEEE/OSA Journal of Lightwave Technology (JLT), vol. 32, pp. 2780-2795, 2014.

[7] A.P. Vela, A. Vía, F. Morales, M. Ruiz, and L. Velasco: *Traffic generation for telecom cloud-based simulation.* IEEE International Conference on Transparent Optical Networks (ICTON), 2016.

[8] OMNeT++ Discrete Event Simulator: http://www.omnetpp.org/

[9] P. S. Barreto, T. F. Bento, Paulo H. P. de Carvalho: *Multimedia Network Simulation with a Fluid Flow Model.* IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[10] C. Kiddle, R. Simmonds, C. Williamson, and B. Unger: *Hybrid Packet/Fluid Flow Network Simulation.* Proceedings of the Seventeenth Workshop on Parallel and Distributed Simulation (PADS?03) 1087-4097/03, IEEE 2003.

[11] Vickrey, W.S: *Congestion theory and transport investment.* The American Economic Review 59 (2), 251261.

[12] Ke Han, Terry L. Friesz, Tao Yao: *A partial differential equation formulation of Vickrey's bottleneck model, part I: Methodology and theoretical analysis.* Transportation Research Part B Methodological. March, 2013.

[13] Ke Han, Terry L. Friesz, Tao Yao: *A partial differential equation formulation of Vickrey's bottleneck model, part II: Numerical analysis and computation.* Transportation Research Part B Methodological. March, 2013.

[14] S.M. Ross: *Introduction to Probability Models, Eleventh Edition.* Academic Press. February, 2014.

[15] E. A. Coddington, N. Levinson: *Theory of Ordinary Differential Equations.* New York: McGraw-Hill.

[16] Arieh Iserles: *A First Course in the Numerical Analysis of Differential Equations* Cambridge Texts in Applied Mathematics, 2nd Edition.

[17] Michael Spivak: *Calculus.* Publish or Perish, 4th edition.