Introduction

"What began as an exercise in military paranoia has become a method of global communication" [2].

The Internet is a rapidly-growing worldwide network of computer networks that connects millions of users in more than 170 countries. The recent exponential growth of the Internet poses challenging problems in terms of control of the network. One of the basic concepts which contributed to its rapid growth and popularity is the so-called "end-to-end argument". The idea behind this argument is: keep the network simple and put the intelligence in the edges of the network [108, 37, 66]. Even though this concept contributed to the success of the Internet, it made it very difficult to have any information on the internal state of the network.

Whenever a function is to be added to the Internet, one has the choice of adding it either to the network or to applications. Adding a single function to the core of the network is highly difficult as all the routers in the world would have to be updated, which is not very tractable due to the huge size of the Internet.

Another component of the success of the Internet is its heterogeneity and its ability to connect wired to wireless networks, as well as terrestrial to satellite networks. This heterogeneity presents yet another difficulty when a certain function is to be added to the network. For all of these reasons, we believe it is much more attractive to add the required functions to applications.

Description of the subjects studied

In this thesis, we are interested in solving problems related to different types of applications and in each case we will adopt an end-to-end approach.

First, we looked at *unicast* applications wishing to perform congestion control. Applications using TCP (Transmission Control Protocol [98]) as a transport protocol are not concerned with such control as this function is implemented in TCP [67]. Applications using UDP (User Datagram Protocol [97]) will have to control their sending rate in order to avoid congestion and/or to be TCP-friendly [4]. To enable such a control in such applications, one must provide the application with the state of the path connecting the sender to the sink. For instance, the available bandwidth on a path is well suited for congestion control and the

bottleneck bandwidth is well suited for transmission rate control. We propose an end-to-end methodology which supplies the application with estimates on network-internal characteristics triggering some adaptation algorithm. Another kind of application that could benefit from this work is load balancing in routers and capacity planning as our methodology allows one to identify congested and underutilized links.

In the second part of the thesis, we are concerned with *multicast* applications and, more precisely, with the estimation of the membership of multicast sessions. With the emergence of multimedia applications [119] and dedicated sophisticated hardware such as high-quality video and audio cards, one is expecting to see major events and live sports being multicast. It will then be of significant importance to the sender to have an estimate of the number of receivers. This will allow the sender to determine the popularity of emitted content and even to advertise when the number of receivers is high. Not only can the sender benefit from such estimates: Internet Service Providers (ISPs) may take advantage of this information and charge the source based on the number of receivers instead of charging on an input-rate basis. One may think that the easiest way to provide membership estimates to the source would be to implement this function in the network. However, this solution is not convenient for multiple reasons. First, it requires that each multicast router maintain in memory one counter per multicast session, which is clearly not scalable. Second, it requires all of multicast routers to be changed. Third, such estimates will not be precise as gateway routers will have no way of determining the number of receivers within Local Area Networks (LANs), the only information available to them being the possible existence of at least one receiver [43]. Such an approach will rather return a lower bound on the number of receivers. Here again, we believe that an end-to-end approach is more suitable.

The third and last part of the thesis is concerned with applications in a *mobile agent environment*. In such applications, components of a running program can move from one host to another and proceed with their execution on the new host. This is often called the "code mobility paradigm" [117] and is very useful to load balancing. Another possible application is data mining [33] in which the data to be processed is situated on a remote machine. Moving the code to the location of the data will save bandwidth as the code is usually smaller in size. Inherent problems to this paradigm are insuring communications between the different components of the applications. Usually, mobile code libraries supply the applications with communication mechanisms. There are two widely used mechanisms, and we will, in this thesis, analyze one implementation of each. The first mechanism is decentralized and relies on special objects called "forwarders" that will forward the messages to the moving components. The second mechanism is centralized and relies on a location server which records the current positions of all of moving components. The question which arises then is: which mechanism performs better? This Quality of Service (QoS) issue has motivated our work on the subject. The mobile code libraries implementing one or the other mechanism have not justified their choice formally (see for instance [84]), and there is no formal comparison of the performance of both mechanisms.

Thesis contributions

This thesis makes several contributions to the queueing theory, to the networking field and to the code mobility field.

In the first chapter, we propose two queueing models based on the $M/M/1/K$ and $M/D/1/K$ queues, and derive several QoS metrics of interest in each model, such as the loss probability, the server utilization, the response time and some conditional probabilities. For the first time, metrics such as the conditional loss probability and the conditional non-loss probability are derived for the $M+M/M/1/K$ queue. Furthermore, our detailed analysis of the $M/D/1/K$ queue has not been carried out before. Explicit information is provided on how to compute the probability distribution of the queue length, which triggers the computation of the loss probability, the server utilization and the response time of the $M/D/1/K$ queue. Based on these models, we propose several schemes for estimating network-internal characteristics, and evaluate their performance on simulations conducted with `ns-2` [83].

In the second chapter, we embark towards estimating the membership in multicast sessions. We first propose to model the multicast group as an $M/M/\infty$ queue under a heavy traffic regime. In this case, the scaled membership weakly converges to an Ornstein-Ühlenbeck process which is known to have linear dynamics. The fact that both the dynamics of the system and the measurements are linear enables the use of the powerful Kalman filter theory. The estimator derived from this theory is ensured to be optimal under the considered assumptions. The linearization of the problem and the use of the Kalman filter in this context is a very original contribution. Motivated to derive an estimator under more general assumptions, we build on Wiener filter theory which returns the best filter, in steady-state, among all of linear filters. The model considered in this case is the $M/M/\infty$ queue under a general traffic regime. The last estimator we derive in this chapter, is based on an $M/H_L/\infty$ queue model where H_L denotes an L-stage hyperexponential distribution. All of these estimators have been applied on real audio and video traces, and they appear to perform very well despite the fact that the assumptions considered in the different models were violated in the considered traces. Note that it is the first time that membership estimators are applied to real traces as usually they are only tested on synthetic traces.

The material presented in the third chapter is very original as it falls within the intersection of two very different fields: the modeling world and the code mobility world. Researchers from the first field are often not familiar with the problems encountered in the second field, and researchers from the code mobility world are often not familiar with modeling techniques such as Markov chains. This fact has impaired the performance evaluation of mechanisms related to code mobility. We model two implementations of location mechanisms using Markov chains. The forwarders mechanism is modeled as an infinite state-space Markov chain, and the state probability distribution is expressed using z-transforms. Expressions for the average number of forwarders and the average response time of the mechanism are then found. The centralized mechanism is modeled as a finite state-space

Markov chain and the Chapman–Kolmogorov equations are solved numerically. Here also we find an expression for the average response time of the mechanism. After validating both Markovian models through simulations and experiments over both a LAN and a MAN, the expressions of the expected response times returned by both models are compared under a wide variety of conditions, showing that no mechanism is uniformly better than the other, thereby justifying even more this research.

Thesis organization

The following chapters are self-contained, since each problem is treated individually and in sequential. Chapter 1 is concerned with adaptive unicast applications which transmission rate is to be adjusted to avoid congestion. Chapter 2 is concerned with multicast applications in which the sender is interested in tracking the dynamics of the group membership. Chapter 3 is concerned with the performance evaluation and comparison of two agent location mechanisms in a mobile code environment. Chapter 4 summarizes the thesis and discusses some perspectives. A glossary and the bibliography complete the report.

Notation in use

Here are some notes on the notation used in the following. Vectors and matrices are represented by bold letters. The Kendall–Lee notation [72, 78] is used to classify the queueing models. C_n^p will denote a combination of n objects arranged in groups of size p ($n > 0$ and $0 \leq p \leq n$). Row 1 in a table will denote the first line of the table, i.e., the one having the names assigned to the columns. When counting the number of rows, every line is considered even if it is almost empty. Similarly, column 1 will denote the first column as illustrated in this example:

	column 1	column 2	...
row 1			...
row 2			...
⋮	⋮	⋮	⋱