# UTILIZATION-BASED DELAY GUARANTEE TECHNIQUES AND THEIR APPLICATIONS

A Dissertation

by

SHENGQUAN WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2006

Major Subject: Computer Science

# UTILIZATION-BASED DELAY GUARANTEE TECHNIQUES AND THEIR APPLICATIONS

A Dissertation

by

SHENGQUAN WANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

| | |
|---|---|
| Chair of Committee, | Wei Zhao |
| Committee Members, | Riccardo Bettati |
| | Jennifer Welch |
| | Jianxin Zhou |
| Head of Department, | Valerie E. Taylor |

December 2006

Major Subject: Computer Science

# ABSTRACT

Utilization-Based Delay Guarantee Techniques

and Their Applications. (December 2006)

Shengquan Wang, B.S., Anhui Normal University;

M.S., Shanghai Jiao Tong University;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Wei Zhao

Many real-time systems demand effective and efficient delay-guaranteed services to meet timing requirements of their applications. We note that a system provides a *delay-guaranteed service* if the system can ensure that each task will meet its predefined end-to-end deadline. Admission control plays a critical role in providing delay-guaranteed services. The major function of admission control is to determine admissibility of a new task. A new task will be admitted into the system if the deadline of all existing tasks and the new task can be met. Admission control has to be *efficient* and *efficient*, meaning that a decision should be made quickly while admitting the maximum number of tasks.

In this dissertation, we study a *utilization-based admission control* mechanism. Utilization-based admission control makes an admission decision based on a simple resource utilization test: A task will be admitted if the resource utilization is lower than a pre-derived safe resource utilization bound. The challenge of obtaining a safe resource utilization bound is how to perform delay analysis offline, which is the main focus of

this dissertation. For this, we develop utilization-based delay guarantee techniques to render utilization-based admission control both *efficient* and *effective*, which is further confirmed with our data.

We develop techniques for several systems that are of practical importance. We first consider wired networks with the Differentiated Services model, which is well-known as its supporting scalable services in computer networks. We consider both cases of providing deterministic and statistical delay-guaranteed services in wired networks with the Differentiated Services model. We will then extend our work to wireless networks, which have become popular for both civilian and mission critical applications. The variable service capacity of a wireless link presents more of a challenge in providing delay-guaranteed services in wireless networks. Finally, we study ways to provide delay-guaranteed services in component-based systems, which now serve as an important platform for developing a new generation of computer software. We show that with our utilization-based delay guarantee technique, component-based systems can provide efficient and effective delay-guaranteed services while maintaining such advantages as the reusability of components.

# DEDICATION

To MY PARENTS, and MY SISTERS.

# ACKNOWLEDGMENTS

I would like to express my deep appreciation to my advisor, Dr. Wei Zhao, for his introducing me to the field of computer science. I am greatly indebted to him for his constant inspiration, encouragement, patience, and guidance throughout my Ph.D. studies, which were essential to the completion of this dissertation. His broad vision and deep insight have always been the source of inspiration for me in my professional growth.

I would like to thank Dr. Riccardo Bettati, for his great contribution in the development of my research and his involvement on my advisory committee. Throughout my Ph.D. studies, I have benefited tremendously from the intensive interaction with him on many research projects, in particular, at the system level.

Many thanks go to Dr. Jennifer Welch for serving on my advisory committee. I appreciate her deep insight in the area of distributed algorithms.

I would like to thank Dr. Jianxin Zhou for his involvement on my advisory committee, in particular in my final defense.

Appreciation is due to Dr. Martin Wortman for his involvement on my advisory committee in the beginning and his insightful comments on my research proposal.

It is also my great pleasure to acknowledgement many professors and researchers with whom I had the honor to know and work with. Among them, I am especially grateful to Dr. Jianer Chen. His sensitivity on research and sincerity on teaching have been a great source of inspiration. I would like to thank Mr. Willis Marti for his support

for my graduate assistantship, Dr. Donald Friesen for his kind advice relating to many academic issues, and Dr. Walter Daugherity, Dr. Du Li, Dr. Andreas Klappenecker, and Dr. Steve Liu for their friendship and encouragement.

Many thanks also go to my fellow former and current graduate students in the Real-Time Systems group. I have benefited greatly from working with Dr. Dong Xuan, Mr. Zhibin Mai, Mr. Ripal Nathuji, Dr. Nan Zhang, and Dr. Sangig Rho. I have also had many helpful discussions with Dr. Chengzhi Li, Dr. Byung-kyu Choi, Dr. Yong Guan, Dr. Xinwen Fu, Dr. Shu Jiang, Dr. Soohyun Cho, Dr. Ye Zhu, Mr. Bryan Graham, Mr. Wei Yu, and Mr. Youngwoo Ahn.

I would like to express my appreciation to Ms. Elena Catelena, Ms. Larisa Archer, and Ms. Kathy Flores for their friendship and help during my Ph.D. studies. In particular, I would like to thank Ms. Archer for her selfless help in polishing most of my writings.

Finally, I owe a special debt of gratitude to my parents and my sisters. Without their tremendous sacrifices and great love, I would not have the chance to pursue an advanced education, i.e., my graduate studies abroad.

# TABLE OF CONTENTS

Page

Page

Page

# LIST OF FIGURES

Page

Page

# LIST OF TABLES

Page

# CHAPTER I

# INTRODUCTION

## 1. Overview

### 1.a. Delay-Guaranteed Service

Many real-time systems, such as Voice over IP, military command and control systems, and industrial control systems, demand effective and efficient delay-guaranteed services to meet timing requirements of their applications. We note that a system provides a *delay-guaranteed service* if the system can ensure that each (computational or communication) task will meet its predefined end-to-end deadline [1].

For instance, in telecommunication systems, Voice over IP is a typical real-time application [2]. Customers can use IP telephones to communiacte with one another via the Internet, a service is much cheaper than the traditional telephone services. As more customers recognize the benefit, particularly during busy calling periods. When many customers use the service at the same time, phone calls will compete for link bandwidth, cause network congestions, and consequently experience long delays. Once delays exceed a certain threshold, the quality of calls will become intolerable. In industrial/military command and control systems, radar signal processing and tracking is another typical real-time application [3]. In these types of systems, some processors will sample and digitize the echo signal from radars. Then some processors will analyze the data, interface with the display system and also generate command to control the radars,

---

This dissertation follows the style and format of *IEEE/ACM Transactions on Networking*.

in order to track objects in its coverage. Because it is a mission-critical application, timing requirement is extremely important.

### 1.b. Admission Control

In order to provide delay-guaranteed services for real-time applications, we have to introduce into real-time systems resource management schemes such as admission control, packet scheduling, and other signaling protocols. Admission control is one of the most important of these. The major function of admission control is to determine admissibility of a new task. A new task will be admitted into the system if the deadline of all existing tasks and the new task can be met. Admission control mechanisms have been investigated extensively in literature [4], [5], [6], [7], [8], [9], [10]. Among them are two major mechanisms: *delay-based admission control* and *utilization-based admission control*. The *delay-based admission control* mechanism [4], [5], [6] is an approach that performs delay tests at admission time: For each task admission request, the admission control needs to explicitly compute delays for all existing tasks and the new task to check whether delay-guaranteed services can be met or not. In *utilization-based admission control*, the utilization is defined as the portion of resource on average. The utilization-based admission control mechanism [7], [8], [9], [10] makes an admission control decision based on a pre-defined safe resource utilization bound: For each task admission request, as long as the used resource utilization plus the requested resource utilization are not beyond the pre-defined safe resource utilization bound that is computed offline, the service guarantee can be provided.

Rapidly growing workload demands and the continuous growth of the system size in real-time systems are placing enormous processing demands in admission control, in particular within the systems burdened with numerous of tasks. There are two primary goals in the design of admission control mechanisms for such kind of systems. On the one hand, admission control has to be *efficient*, meaning that an admission decision should be made as quickly as possible, requiring the admission control mechanism to be lightweight. On the other hand, admission control should be *effective*, meaning that a system should admit as many tasks as possible in order to achieve high resource utilization. Delay-based admission control and utilization-based admission control are quite different in achieving these two goals:

- Delay-based admission control is not efficient. The admission control needs to explicitly compute delays of *all existing tasks and the new task* at the admission time, which is usually computationally expensive in systems with a large number of tasks. Why is it necessary to compute the delays for all existing tasks? As we know, once a new task is admitted into the system, it will interfere with many existing tasks since they compete for the same resource. In the worst-case, all existing tasks can be affected. Therefore, the admission control has to ensure that all existing tasks will still meet their deadline requirements once the new task is admitted. Despite the fact that it is not efficient, delay-based admission control is very effective.
- Utilization-based admission control involves only a simple utilization test and eliminates the delay computation at the admission time. Utilization-based

admission control renders the admission control very efficient. However, it tends not to be effective, since it does not take into account of the dynamics of tasks in the admission process.

In our work, we adopt the utilization-based admission control mechanism; its efficiency renders it very suitable for systems loaded with a large number of tasks. In this dissertation, we intend to show that utilization-based admission control can be both efficient and effective with our proposed utilization-based delay guarantee techniques.

## 2. Utilization-Based Delay Guarantee Techniques and Their Applications

Resource management is essential to providing delay-guaranteed services. Different systems may have different kinds of resources or different characteristics of resources. For instance, in wired networks, the key resource is link bandwidth and the link bandwidth is constant. In wireless networks, the key resource is still link bandwidth, but the link bandwidth might be variable. In software component systems, the key resource will not be link bandwidth, but CPU cycles.

In this dissertation, we will focus on development of utilization-based delay guarantee techniques for these practical systems.

1) We first consider deterministic delay-guaranteed services in wired networks with the Differentiated Service model. The Differentiated Service model is well-known for its advantage of being able to support scalable services for aggregated traffic in computer networks. We develop a utilization-based delay

guarantee technique to provide hard delay-guaranteed services for real-time application in these systems.

2) We then extend the work on the Differentiated Services model from deterministic delay-guaranteed services to statistical delay-guaranteed services. Statistical delay-guarantee recognizes the fact that many applications can tolerate a small percentage of missing deadlines, hence rendering the utilization-based admission control more effective than the deterministic one.

3) We also extend our work from wired networks to wireless networks. Wireless networks continue to experience a surge in popularity for both civilian and mission critical applications. However, the capacity of a wireless link may vary dramatically, making delay-guarantee particularly challenging. We address how to develop a utilization-based delay guarantee technique in wireless networks.

4) Finally, we study how to provide delay-guaranteed services in component-based systems, which now serve as an important platform for developing a new generation of computer software. We show that with our utilization-based delay guarantee technique, component-based systems can efficiently and effectively provide delay-guaranteed services while maintaining their other advantages such as reusability of components.

## 3. Organization of this Dissertation

The rest of this dissertation is organized as follows: In Chapter II, we review the related work. In Chapter III, we focus on developing a utilization-based delay guarantee technique to provide deterministic delay-guaranteed service in wired networks with the Differentiated Services model. In Chapter IV, we consider to extend the work in Chapter III to statistical delay-guaranteed services. How to provide delay guarantees in wireless networks is addressed in Chapter V. In Chapter VI, we study ways to provide delay-guarantees in component-based systems. Finally, we conclude this dissertation research with a brief summary in Chapter VII.

# CHAPTER II

# RELATED WORK

## 1. Delay-Guaranteed Techniques

Developing delay-guaranteed techniques are always the central focus of the real-time systems community and also in our research group. In [11], Zhao studied a heuristic approach to scheduling hard real-time tasks with resource requirements in distributed systems. In [12], Bettati presented the end-to-end scheduling to meet deadlines in distributed systems. Li [13] concentrated on the techniques for the analysis of the network traffic and its applications in the network management. Choi [14] focused on resource management for scalable Quality of Service. Wu [15] performed general schedulability bound analysis in real-time systems.

In our work, we adopt utilization-based admission control mechanism to provide delay-guaranteed services in real-time systems. In its basic form, utilization-based admission control was first proposed [16] for preemptive scheduling of periodic tasks on a simple processor. A number of utilization-based tests are known for (i) centralized systems such as 69% (and the extensions) for the Rate/Deadline Monotonic scheduler [16], [17], [18], [19], [20], [21], 100% for Earliest Deadline First scheduler [16], and the general schedulability bound analysis for static-priority scheduler [15], [22], (ii) multiprocessor parallel systems including some important utilization bound results [23], [24], [25], [26], or (iii) distributed systems such as 33% for the Timed Token protocol over Fiber Distributed Data Interface (FDDI) networks [27], [28], [29], [30].

## 2. Delay Guarantees in Wired Networks with the Differentiated Services Model

A good survey on recent work in Differentiated Services has been illustrated in [31]. Nichols et al. [32] proposed the *premium service* model, which provided the equivalent of a dedicated link between two access routers. It provided Differentiated Services in priority-driven scheduling networks with two priorities, in which the high priority was reserved for premium service. The algorithm in [33] provided both guaranteed and statistical rate and delay bounds, and addressed scalability through traffic aggregation and statistical multiplexing. Stoica and Zhang [34] described an architecture to provide guaranteed service without per-flow state management by using the Dynamic Packet State (DPS) technique. However, the packet headers had to be changed to implement their proposed technique. Our work is based on the static-priority scheduling algorithm that is relatively simple and widely supported.

The utilization-based admission control mechanism is not new to networks. The fluid-flow model in the Integrated Services model, for example, allowed various forms of utilization based admission control [35]. Such approaches cannot be used in a Differentiated Services model, however, because they rely on guaranteed-rate schedulers, which need to maintain the flow information. A utilization-based delay analysis has been studied most recently in [36] for the case of aggregate scheduling. Lower bounds on the worst-case delay have been derived. These bounds are a function of the network utilization, the maximum hop count of any flow, and shaping parameters at the entrance to the network. However, only First-In First-Out (FIFO) scheduling is under consideration. Also, delay bounds are not tight, but independent of the network

topology. In our work, we derive a better delay bound in static-priority scheduling networks. A similar work on the utilization-based delay guarantee technique was also studied in [7], where it was assumed that priorities were assigned on a class-by-class basis. In our work, by relaxing this assumption, the priority assignment is not constrained by class membership and flows from a class can be assigned different priorities. The problem presented in our work is more general and challenging. While utilization-based admission control significantly reduces the admission control overhead, excessive flow establishment activity can still add substantial strain to the admission control components. In [9], an endpoint admission control mechanism was proposed to reduce resource allocation overhead at the admission time by appropriately pre-allocating resources. This mechanism supplements our work.

Statistical delay-guaranteed service has been studied via different statistical envelopes, such as envelopes of bounding moment generating functions [37], exponentially bounded envelopes [38], [39], and envelopes consisting of families of bounding distributions [40], [41]. Statistical envelopes were also applied to resource allocation for inter-class resource sharing [42] and video-on-demand services [43]. Much work has been done to generalize schedulability conditions for a deterministic service to a statistical framework. Several researchers made statistical extensions to deterministic service models. In [44], a rate-variance envelope was introduced, which described the variance of the arrivals of a flow as a function of a time period of length. In [45], arrivals on a flow were assumed to be characterized by the rate-variance envelope and a long-term arrival rate. Then, applying the central limit theorem argument, a bound for the

probability of a delay bound violation was derived for a static-priority scheduler. In [46], the authors used a rate-variance envelope as a simple way to capture the second-moment properties of temporally correlated traffic flows, and to describe how quickly the rate-distribution becomes concentrated at the mean rate with increasing interval-length (a key factor for computing deadline violation probabilities). However, these approaches can not be applied to the Differentiated Services framework, since their delay computation relies on the run-time information of flow distribution. Our work intends to solve this problem.

## 3. Delay Guarantees in Wireless Networks

The difficulty of provisioning delay-guaranteed services in wireless networks stems from the need to explicitly consider both the wireless channel transmission characteristics and the underlying error control mechanisms. There is a large volume of literature dealing with the representation and analysis of wireless channel models, and most of these models directly characterize the fluctuations of signals and provide an estimate of the performance characteristics, such as symbol error rate vs. signal-to-noise ratio [47]: The classical two-state Gilbert-Elliott model [48], [49] for burst noise channels, which characterized error sequences, had been widely used and analyzed. In [50], a multiple-state quasi-stationary Markov channel model was used to characterize the wireless non-stationary channel. In [51], [52], a finite-state Markov channel was illustrated with multiple states representing the reception at different signal-to-noise levels. A fluid version of the Gilbert-Elliott model was used in [53] to perform delay

analysis and packet-discard performance as well as the effective capacity for service guarantee support over a wireless link with automatic repeat request (ARQ) and forward error correction (FEC).

In our work, our utilization-based delay guarantee technique can be applied to any of these wireless link models. At the same time, we do not assume a particular traffic pattern, such as the ON/OFF traffic model used in [53]. Instead, we use rate-variance envelopes [44], a simple and general traffic characterization. This methodology makes our approach applicable to any particular situation.

## 4. Delay Guarantees in Component-Based Systems

Component standards specify widely-accepted interfaces that allow independent components from different suppliers (third parties) to be plugged together and to interoperate across language, compiler, and platform barriers. The best known examples of such standards are OMG's CORBA [54], Microsoft's COM+ [55], and Sun Microsystems' Enterprise JavaBeans (EJB) [56]. Although component-based models deal successfully with functional attributes, they provide little support for real-time services. Existing standards – such as CORBA, COM+, and EJB – are unsuitable for real-time applications because they do not address issues of timeliness and predictability of service, which is a basic requirement of real-time systems [57]. The Real-Time, Embedded, and Specialized Systems (RTESS) Platform Task Force of the OMG proposed a specification for a real-time CORBA [58], [59]. At the same time, there is no specification for a real-time EJB or a real-time COM+ yet. The TAO project [60]

provided a CORBA implementation that guaranteed that tasks across components preserve priority levels and that overhead in servicing a task request was statically predictable. This made the ensuing system amenable to the same static schedulability analysis used in traditional real-time systems. After the architectural design phase was completed, a TAO solution would be considered for the AOCS software. In [61], an approach was proposed for run-time fast software component migration (lightweight migration and proactive resource discovery) for application survivability in distributed real-time systems. In [62], the authors proposed to use component-based techniques for developing embedded systems software, i.e., software for resource-constrained systems, and the VEST toolkit aimed at providing a rich set of dependency checks based on the concept of aspects to support distributed embedded system development via components.

However, most of these real-time extensions use traditional approaches to provide delay-guaranteed services, without addressing the reusability in terms of both functional and delay-guaranteed services. Our work intends to solve this problem.

# CHAPTER III

# DETERMINISTIC DELAY GUARANTEES IN WIRED NETWORKS WITH THE DIFFERENTIATED SERVICES MODEL*

In the following two chapters, we will develop utilization-based delay guarantee techniques in wired networks with the Differentiated Services model. In this chapter, we first focus on providing *deterministic* delay-guaranteed services for real-time applications in such kind of systems.

## 1. Overview

In computer networks, a flow is a communication task. Traditionally, architectures for providing delay guarantees over computer networks rely on detailed per-flow information. In the IETF Integrated Services architecture [64], for example, each flow is controlled both by admission control at admission time and by packet scheduling during the flow lifetime. At the flow establishment time, the necessary resources must be allocated to the new flow if the admission request is approved. During the flow lifetime, the flow is policed to ensure that the abnormal behavior of a flow does not affect other flows. This necessitates that information about each flow is kept by each node along the path for admission control and packet forwarding.

It is agreed upon that Integrated Services do not scale. High-speed routers are required to maintain state and perform scheduling decisions for large numbers of flows.

---

In addition, as the number of flows increases, the run-time overhead incurred in flow establishment and tear-down increases as well. The Integrated Services architecture therefore cannot provide scalable Quality-of-Service (QoS) guaranteed services (such as delay-guaranteed services), and the lack of scalability is due to overhead, both at the flow establishment time and during the flow lifetime.

The Differentiated Services model is a proposed standard aimed at supporting scalable services over the Internet through *aggregation of flows* into *service classes* [65]. Network nodes in Differentiated Services need not maintain per-flow information. Since each router only guarantees that service agreements are *locally* maintained on a per-class basis, end-to-end guarantees are difficult to provide. In this chapter, we will focus on wired networks with the Differentiated Services model that uses the utilization-based admission control mechanism to provide end-to-end delay-guaranteed services.

The challenge of using the utilization-based admission control mechanism in the Differentiated Services framework is how to verify whether a utilization bound is safe in providing delay-guaranteed services at the system configuration time. Obviously, the verification will have to rely on a delay analysis method. We will follow the approach proposed by Cruz [66] for analyzing delays. Cruz's approach must be adapted to be applicable in a flow-distribution-unaware environment however. The delay analysis proposed in [66] depends on the information about flow distribution, i.e., the number of flows at input links and the traffic characteristics (e.g., the average rate and burst size) of flows. In our case the delay analysis is done at the system configuration time when the information on flow distribution is not yet available. We will develop a utilization-based

delay guarantee technique that allows us to analyze delays without depending on the dynamic information about flow distribution.

We also assume that real-time applications have the deterministic deadline requirement (we will extend our work into statistical guarantee in the next chapter). We also use *static-priority* schedulers at all routers. Static-priority scheduling algorithm is widely supported on the Internet. The most important feature of this scheduling algorithm is that it is very efficient. Since static-priority scheduling only maintains priorities information, it has little overhead, which is very important in the networks loaded with a large number of flows.

## 2. Models

### 2.a. Network Models

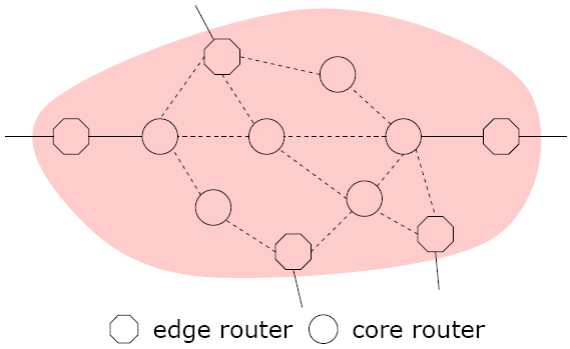



Figure III-1. Network Model

In our Differentiated Services architecture, there are two types of routers (as shown in Figure III-1): *Edge routers* are located at the boundary of the network and provide

support for traffic policing (e.g., leaky bucket); *Core routers* are inside the network. A router is connected to other routers, or hosts, through its input and output links. For the purpose of delay computation, we follow standard practice and model a router as a set of *servers*, one for each router component, where packets can experience delays. Packets are typically queued at the output buffers, where they compete for the output link bandwidth. We therefore model a router as a set of *output link servers*. All other servers (input buffers, non-blocking switch fabric, wires, etc.) can be eliminated from the delay analysis by appropriately subtracting constant delays that have been incurred from the deadline requirements of the traffic. Consequently, the network can be modeled as a graph where the output link servers[1] are nodes and are connected through either links in the network or paths within routers, which both make up the set of edges in the graph. We assume Server $k$ is of capacity $C_k$ and has $L_k$ input link servers with capacities $C_{j,k}, j = 1,\ldots, L_k$.

## 2.b. Traffic Models

We call a stream of packets between a sender and a receiver a *flow*. Packets of a flow are transmitted along a single *flow route*, which we model as a sequence of servers. Following the Differentiated Services model, flows are partitioned into classes. QoS requirements and traffic specifications of flows are defined on a class-by-class basis. We use $M$ to denote the total number of classes in the network. We assume that at each server, a certain portion of bandwidth is reserved for each class traffic separately. Let $\alpha_k^i$

[1] In the following, if the context is clear, we will use link server or server to refer to output link server.

denote the portion of bandwidth reserved for class-$i$ traffic at Server $k$. We assume static-priority schedulers with support for $P$ distinct priorities in the routers. The bandwidth assigned to class-$i$ traffic at Server $k$ is further partitioned into portions $\alpha_{p,k}^{i}$, one for each class-$i$ traffic with priority $p$ at that server. We note that $\alpha_{k}^{i} = \sum_{q=1}^{P} \alpha_{q,k}^{i}$ (the question of how much bandwidth to assign to each priority traffic will be discussed in Section 6).

In order to appropriately characterize traffic both at the ingress router and within the network, we use a general traffic descriptor in form of traffic functions and their time-independent counterpart, constraint traffic functions [66]:

**Definition III-1.** For a traffic flow, its traffic at a specific point is characterized by a *traffic function f(t)* which is defined as the amount of the traffic of the flow passing through the point during time interval [0,t). The function *F(I)* is called the *traffic constraint function* of *f(t)* if

$$f(t+I) - f(t) \le F(I), \tag{III-1}$$

for any $t>0$ and $I>0$. This definition is also applied to aggregated traffic flows.

We assume that the source traffic of a flow in Class $i$ is controlled by a leaky bucket with burst size $\sigma^{i}$ and average rate $\rho^{i}$. Define $H^{i}(I)$ as the *source traffic constraint function* for any class-$i$ traffic flow, which is constrained at the entrance to the network by

$$H^{i}(I) \le \sigma^{i} + \rho^{i} I. \tag{III-2}$$

A leaky bucket and the traffic constraint function for a shaped class-$i$ flow are illustrated in Figure III-2.

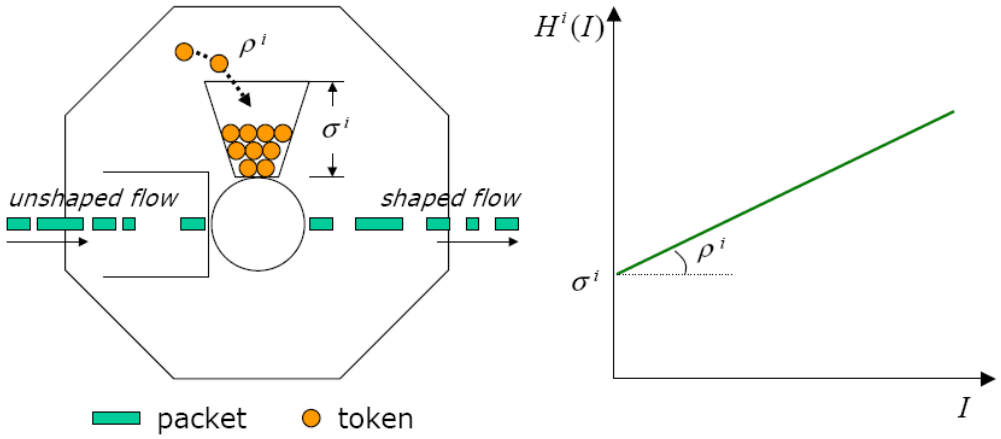



Figure III-2. A Leaky Bucket in Edge Router and the Traffic Constraint Function for a Shaped Class-$i$ Flow

Since the QoS requirement of traffic (in our case, end-to-end deadline) is specified on a class-by-class basis, we can use characteristic parameters $\langle \sigma^i, \rho^i, D^i \rangle$ to represent class-$i$ traffic, where $D^i$ is defined as the end-to-end deadline requirement of any class-$i$ packet. A queue is stable in the sense of bounded queue length and bounded delay for customers if the long-term average rate of the input traffic is smaller than the capacity [37]. A network is said to be stable if all the data packets experience bounded delays within the network [69]. As long as the utilization of all individual links is less than $100\%$ (this can be achieved by admission control), the stability can be guaranteed [70]. In this work, we perform the delay analysis in a stable network. We use $d_{p,k}$ to denote the local worst-case delay suffered by any packet with priority $p$ at Server $k$.

## 3. A QoS Architecture for Differentiated Services

In this section, we propose an architecture to provide delay-guaranteed services in wired networks with the Differentiated Services model and static-priority schedulers. This architecture consists of three major modules:

- *Utilization Bound Verification:* In order to allow for a utilization-based admission control to be used at the admission time, safe utilization bounds at all servers must be determined during the system configuration time. Using a utilization-based delay computation method, a delay upper bound is determined for each priority traffic at each server. This module then verifies whether the end-to-end delay bound in each feasible flow route of the network satisfies the deadline requirement, as long as the bandwidth usage on the flow route is within a pre-defined threshold – the safe utilization bound. This is also the point when priorities are assigned within classes and when bandwidth is assigned to classes and to priorities. We will discuss bandwidth and priority assignment algorithm later.
- *Utilization-Based Admission Control:* Once safe utilization levels have been verified at the system configuration time, the admission control only needs to check whether the necessary bandwidth is available along the flow route of the new flow. Once a new flow arrives, each server along the flow route will be checked to see whether there is sufficient bandwidth available. If this is satisfied at all servers along the flow route, then the new flow will be admitted and the available bandwidth of all servers along the flow route will be

decreased by the requested bandwidth; otherwise, the new flow will be blocked. Once the flow terminates, the bandwidth requested by the flow will be released at each server along the flow route.

- *Packet Forwarding:* In a router, packets are transmitted according to their priorities, which can be derived from the (possibly extended) class identifier in the header. Within the same priority, packets are served in FIFO order.

Out of the above three modules, the first module, utilization bound verification, is the most important. Utilization-based admission control works around a utilization bound. Whether this utilization bound is safe or not needs to be verified at the system configuration time. We propose to realize this critical function in three steps: (i) We first obtain a general delay formula which depends on dynamic flow information. (ii) We then remove this dependence to obtain a utilization-based delay formula. (iii) Finally, we verify whether or not this utilization bound is safe by applying the utilization-based delay formula.

Of the three steps listed previously, Step (iii) is relatively straightforward once we have the utilization-based delay formula. We can apply the delay formula to check, under the given utilization bound, whether the end-to-end delay bound in each feasible flow route satisfies the deadline requirement. Accordingly, we will focus on Step (i) and Step (ii).

# 4. Utilization-Based Delay Analysis

In this section, we will present a new utilization-based delay computation formula, which is independent of the dynamic information of flow distribution.

## 4.a. Main Result

Since static-priority scheduling does not provide flow separation, the local delay at a server depends on detailed information (number and traffic characteristics) of other flows both at the server under consideration and at servers upstream. Therefore, all the flows *currently* established in the network must be known in order to compute delays. Delay formulas for this type of system have been derived for a variety of scheduling algorithms [67]. While such formulas could be used (albeit expensively) for flow establishment at the admission time, they are not applicable for delay computation during the system configuration time, as they rely on the dynamic information of flow distribution.

In the absence of such information, the worst-case delays must be determined assuming a worst-case combination of flows. Fortunately, the following theorem gives an upper bound on this worst-case delay without having to exhaustively enumerate all kinds of flow distributions. In the following discussion, we will rely heavily on the following vector notation: If the symbol $a^i$ denotes some value specific to class-$i$ traffic, then the notation $\vec{a}$ denotes an $M$-dimensional vector $(a^1, a^2, ..., a^M)$. We will use the operator "$\cdot$" for the inner product and the operator "$\|\cdot\|$" for the vector norm, i.e., $\vec{a} \cdot \vec{b} = \sum_{i=1}^{M} a^i b^i$, $\|\vec{a}\| = \sum_{i=1}^{M} |a^i|$.

**Theorem III-1**. The worst-case delay $d_{p,k}$ suffered by any packet with priority $p$ at Server $k$ can be bounded as

$$d_{p,k} \le \frac{1}{\left\|\vec{\alpha}_{p,k}\right\|} \sum_{q=1}^{p} \varpi_{q,k} (\vec{\alpha}_{q,k} \cdot \vec{Z}_{q,k}), \tag{III-3}$$

where

$$\varpi_{q,k} = \begin{cases} 1, & q < p \\ \dfrac{c_k - \left\|\vec{\alpha}_{q,k}\right\|}{c_k - \left\|\alpha_{q,k}\right\|} & q = p \end{cases}, \tag{III-4}$$

$$\left\|\vec{\alpha}_{p,k}\right\| = 1 - \sum_{q=1}^{p-1} \left\|\alpha_{q,k}\right\|, \tag{III-5}$$

$$c_k = \frac{1}{C_k} \sum_{j=1}^{L_k} C_{j,k}, \tag{III-6}$$

$$Z_{q,k} = \frac{\sigma^i}{\rho^i} + Y_{q,k}^i, \tag{III-7}$$

$$Y_{q,k}^i = \frac{\sigma^i}{\rho^i} + \max_{R \in S_{q,k}^i} \sum_{s \in R} d_{q.s}, \tag{III-8}$$

and $S_{q,k}^i$ is the set of all sub-routes used by class-$i$ traffic with priority $q$ upstream from Server $k$. The exception is that $d_{p,k} = 0$ as $c_k = \left\|\vec{\alpha}_{p,k}\right\| = 1$ or $\left\|\vec{\hat{\alpha}}_{p,k}\right\| = 0$.

Derivation of (III-3) will be discussed in Section 5. At this point, we would like to make the following observations about **Theorem III-1**:

- Usually a delay suffered at a server would depend on the state of the server, i.e., the number of flows that are admitted and pass through the server. We note that (III-3) is independent from this kind of information. The values of $\sigma$, $\rho$,

$\alpha_{q,k}$, and $c_k$ are available at the time when the system is (re-)configured. Hence, the delay computation formula is insensitive to the flow distribution information.

- We define $\frac{\sigma^i}{\rho^i}$ as the *relative burstiness* of class-$i$ traffic. The relative burstiness of class-$i$ traffic is the time required for class-$i$ traffic to get to burst size $\sigma^i$ at the average rate $\rho^i$. The delay formula depends on the *relative burstiness* and the bandwidth allocation. As $\frac{\sigma^i}{\rho^i}$ or $\alpha_{q,k}^i$ increases, $d_{p,k}$ increases.

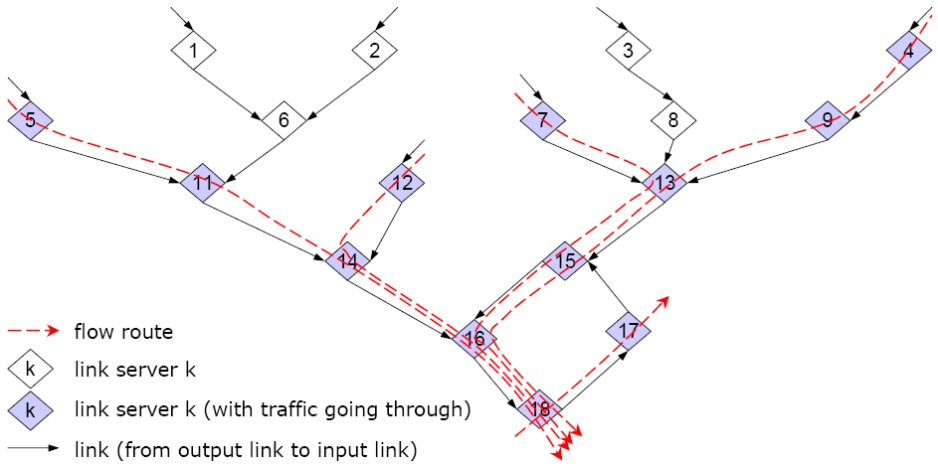



Figure III-3. Computation of $Y_{q,k}^i$

- We note that $d_{p,k}$ in (III-3) depends on $Y_{q,k}^i$. By (III-3), $Y_{q,k}^i$ is the maximum of the worst-case aggregated delays experienced by all class-$i$ packets with priority $q$ upstream from Server $k$ (see Figure III-3). Consider any class-1

traffic with priority 1 in the network as shown in Figure III-3. There are 4 flow routes going through Server 16 ($5 \to 18$, $12 \to 18$, $7 \to 18$, and $4 \to 18$), therefore $Y_{1,16}^1 = \max\{d_{1,5} + d_{1,11} + d_{1,14}, \quad d_{1,12} + d_{1,14}, \quad d_{1,7} + d_{1,13} + d_{1,15}, \quad d_{1,4} + d_{1,9} + d_{1,13} + d_{1,15}\}$. The value of $Y_{q,k}^i$, in turn, depends on the delays experienced at some servers other than Server $k$. Then we have a circular dependency. Hence, the delay values depend on each other and must be computed simultaneously. Define $V$ as the set of all link servers in the network. Recall that there are $P$ available priorities in total, then we use the $(P \times |V|)$-dimensional vector $\vec{d}$ to denote the upper bounds of the delays suffered by the traffic with all priorities at all servers:

$$\vec{d} = (d_{1,1}, d_{1,2}, \cdots, d_{1,V}, d_{2,1}, d_{2,2}, \cdots, d_{2,|V|}, \cdots, d_{P,1}, d_{P,2}, \cdots, d_{P,|V|}). \tag{III-9}$$

Define the right hand side of (III-9) as $\Phi_{p,k}(\vec{d})$, and define

$$\begin{aligned} \Phi(\vec{d}) = (&\Phi_{1,1}(\vec{d}), \Phi_{1,2}(\vec{d}), \cdots, \Phi_{1,|V|}(\vec{d}), \\ &\Phi_{2,1}(\vec{d}), \Phi_{2,2}(\vec{d}), \cdots, \Phi_{2,|V|}(\vec{d}), \cdots, \\ &\Phi_{P,1}(\vec{d}), \Phi_{P,2}(\vec{d}), \cdots, \Phi_{P,|V|}(\vec{d})). \end{aligned} \tag{III-10}$$

The queuing delay bound vector $\vec{d}$ can then be determined by iteratively solving the following vector equation:

$$\vec{d} = \Phi(\vec{d}). \tag{III-11}$$

Once the worst-case delay bound for any packet with any priority at each server is obtained, the end-to-end worst-case delay for packets with priority $p$ on flow route $R$ can be computed as $d_{p,R}^{e2e} = \sum_{k \in R} d_{p,k}$.

The worst-case delay bound suffered by a packet with specific priority is only affected by the traffic with equal or higher priorities.[2] The best-effort traffic has the lowest priority, and will not affect the delay suffered by any real-time traffic, which has higher priorities.

4.b. Special Cases

In some special cases, delay formulas independent of network topology can be derived. This is the case, for example, in a network with a single real-time class traffic assigned a single priority in a network of identical link servers and identical allocations of bandwidth to the class on all servers. In this case, we simplify the notation to let $\sigma = \sigma^1, \rho = \rho^1, Y_k = Y_{1,k}^1, C = C_k, \alpha = \alpha_{1,k}^1$ and $L = \max_k \{L_k\}$. If we loosen the bound on $Y_k$, we will have an explicit delay formula as shown in the following corollary:

**Corollary III-1.** Let $d$ be the maximum of worst-case delays suffered by all real-time class packets across all link servers in the network. If

$$\alpha < \frac{1}{1 + (h-2)(1-\frac{1}{L})}, \tag{III-12}$$

then

[2] Here we ignore blocking due to non-preemption of packets.

$$d \le \frac{1}{\frac{1}{r}-(h-1)}\frac{\sigma}{\rho}. \tag{III-13}$$

Therefore, the end-to-end delay $d^{e2e}$ can be bounded by

$$d^{e2e} \le \frac{h}{\frac{1}{r}-(h-1)}\frac{\sigma}{\rho}, \tag{III-14}$$

where

$$r = \alpha\frac{L-1}{L-\alpha}, \tag{III-15}$$

and $h$ is the length of the longest flow route in the network.

These delay formulas do not depend on the network topology except for the length $h$ of the longest flow route. We note that a very similar result was derived using a different approach in [36]. The proof of **Corollary III-1** is given in Appendix C.

In a given network, if we represent flows between servers as directed links between servers and there are no loops in the resulting graph, this network is called *feedforward* network [40]. If a link server is $k-1$ hops away from the farest source in the resulting graph, we define it as layer-$k$ link server. In Figure III-3, there are $5$ flow routes, $5 \to 18$, $12 \to 18$, $7 \to 18$, $4 \to 18$, and $18 \to 17$. The longest flow route is $4 \to 18$, thus $h = 6$. This network is feedforward since there is no loop in the resulting directed graph. We find that Servers 4, 5, 7, and 12 are all at Layer 1, Servers 8 and 11 are both at Layer $2$, Servers 13 and 14 are both at Layer $3$, and Servers 15, 16, 18, and 17 are at Layer 4, 5, 6, and 7, respectively.

In a *feedforward* network, traffic is progressively pushed forward along the directed links though the network without loops; therefore, we can get a tighter delay bound that is independent of the network topology as follows:

**Corollary III-2.** Let $\hat{d}_k$ be the maximum of worst-case delays suffered by any real-time class packet at layer-$k$ link servers, then we have the following delay bound:

$$\hat{d}_k \le r(r+1)^{k-1}\frac{\sigma}{\rho}, \tag{III-16}$$

and the end-to-end delay $d^{e2e}$ can be bounded by

$$d^{e2e} \le ((r+1)^{\hat{h}}-1)\frac{\sigma}{\rho}, \tag{III-17}$$

where $r$ is defined in (7) and $\hat{h}$ is the number of layers in the feedforward network ($\hat{h} \ge h$).

Generally, if $\hat{h}$ is not much greater than $h$, the delay bound in (9) is much tighter than the bound in (6). In Figure III-3, $h=6, \hat{h}=7, L=3$, if $\sigma = 640$ bits, $\rho = 32{,}000$ bps, $\alpha = 20\%$, then $\frac{1}{1+(h-2)(1-\frac{1}{L})} = 0.27$, $((r+1)^{\hat{h}}-1)\frac{\sigma}{\rho} = 0.031$ s and $\frac{h}{\frac{1}{r}-(h-1)}\frac{\sigma}{\rho} = 0.060$ s. The proof of **Corollary III-2** is given in Appendix D.

## 5. Deriving the Delay Formula

In this section, we discuss how to derive the delay formula given in (III-3). We will start with a formula for delay computation that depends on flow distribution, which we call *general delay formula*. We will describe how to remove its dependency on information about flow distribution.

5.a. General Delay Formula

We aggregate flows into *groups*. All class-$i$ flows with priority $p$ going through Server $k$ from Input Link $j$ form the group $G^i_{p,j,k}$, and all flows with priority $p$ going through Server $k$ from input link $j$ form the group $G_{p,j,k}$. We use $F^i_{p,j,k}(I)$ and $F_{p,j,k}(I)$ to express the traffic constraint function for group $G^i_{p,j,k}$ and group $G_{p,j,k}$ respectively. The constraint function $F^i_{p,j,k}(I)$ can be formulated as the summation of the constraint functions of individual flows, i.e.,

$$F^i_{p,j,k}(I) = \sum_{x \in G^i_{p,j,k}} F_x(I), \tag{III-18}$$

where $F_x(I)$ is the constraint function for flow $x$ in $G^i_{p,j,k}$. Further, the aggregate traffic of group $G_{p,j,k}$ is constrained by

$$F_{p,j,k}(I) = \min\{C_{j,k} I, \sum_{i=1}^{M} F^i_{p,j,k}(I)\}, \tag{III-19}$$

where $C_{j,k}$ is the capacity of the $j$th input link of Server $k$.

In a stable network, the worst-case delay $d_{p,k}$ suffered by any priority-$p$ packet at Server $k$ can then easily be formulated in terms of the aggregated traffic constraint functions and the service capacity $C_k$ of the server according to [67]:

$$d_{p,k} = \frac{1}{C_k} \max_{I < I_{p,k}} (F_{p,k}(I + d_{p,k}) - C_k I), \tag{III-20}$$

where

$$F_{p,k}(I+d_{p,k})=\sum_{q=1}^{p-1}\sum_{j=1}^{L_k}F_{q,j,k}(I+d_{p,k})+\sum_{j=1}^{L_k}F_{p,j,k}(I), \tag{III-21}$$

$$I_{p,k}=\min\{I>0:\sum_{q=1}^{p}\sum_{j=1}^{L_k}F_{q,j,k}(I)\le C_k I\}. \tag{III-22}$$

(III-20) indicates how long a newly-arriving packet with priority $p$ can be delayed at Server $k$ in a stable network. It describes the maximum worst-case delay suffered by a packet due to delay by higher-priority packets that arrived before or during the time the packet is queued, and same-priority packets queued before the arrival of the packet.

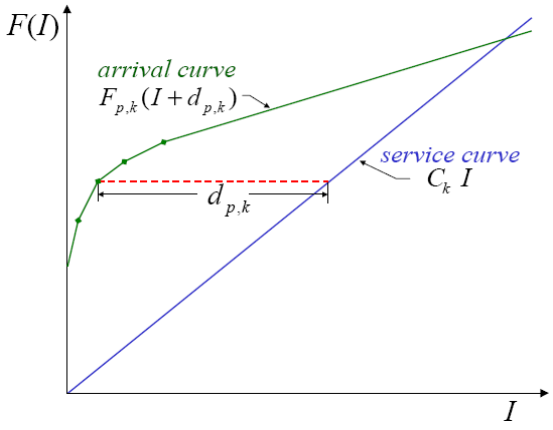



Figure III-4. The Arrival Curve, the Service Curve and the Worst-case Delay.

In (III-20), $F_{p,k}(I+d_{p,k})$ and $C_k I$ form an arrival curve and a service curve, respectively; and $I_{p,k}$ denotes the maximum busy interval of the traffic with priority no lower than $p$ at Server $k$. In other words, Server $k$ never processes packets with priority equal to or higher than $p$ for more than $I_{p,k}$ consecutive time units. The arrival

curve $F_{p,k}(I+d_{p,k})$, the service curve $C_k I$, and the worst-case delay $d_{p,k}$ are illustrated in Figure III-4.[3]

To get the value of $d_{p,k}$, we first need to find the point where the worst-case delay in (III-20) is achieved. This allows us to ignore the maximum operator in (III-20). Assume the arrival curve $F_{p,k}(I+d_{p,k})$ is differentiable almost everywhere for $I>0$. Define $s(I)$ as the piece-wise slope (average rate) of the arrival curve $F_{p,k}(I+d_{p,k})$ at interval $I$, *i.e.*, $s(I)=\frac{\mathrm{d}F_{p,k}(I+d_{p,k})}{\mathrm{d}I}$. As we know, the arrival curve $F_{p,k}(I+d_{p,k})$ is an increasing and concave function. From Figure III-4, we know that the worst-case delay is maximized at the point from which the slope of the aggregate traffic function becomes smaller than $C_k$.

Substituting (III-18) and (III-19) into (III-20), we observe that the above delay formula depends on flow distribution. Suppose that the group $G^i_{p,j,k}$ has $n^i_{p,j,k}$ flows. In fact, (III-20) depends on $n^i_{p,j,k}$, the number of flows in $G^i_{p,j,k}$, and on the traffic constraint functions $F_x(I)$ of the individual flows. This kind of dependency on the dynamic system status must be removed in order to perform delay computations at the system configuration time.

In the following, we describe how we first eliminate the dependency on the traffic constraint functions. Then we eliminate the dependency on the number of flows on each

[3] Here, we slightly abuse the terms “curve” and “function”.

input link. The result is a delay formula that can be applied without knowledge about flow distribution.

5.b. Removing the Dependency on Individual Traffic Constraint Functions

We now show that the aggregated traffic function $F^i_{p,j,k}(I)$ can be bounded by replacing the individual traffic constraint functions $F_x(I)$ with a common upper bound, which is independent of input link $j$.

The delay at each server can now be formulated without relying on traffic constraint functions of individual flows. The following theorem in fact states that the delay for each flow on each server can be computed by using the constraint traffic functions *at the entrance to the network only*.

**Theorem III-2.** The aggregated traffic of the group $G_{p,j,k}$ is constrained by

$$F_{p,j,k}(I) = \begin{cases} C_{j,k} I, \mathrm{I} \le \tau_{p,j,k} \\ n_{p,j,k} \cdot (\eta_{p,k} + \rho I), \mathrm{I} > \tau_{p,j,k} \end{cases} \quad \text{(III-23)}$$

where

$$\tau_{p,j,k} = \frac{n_{p,j,k} \cdot \eta_{p,k}}{C_{j,k} - n_{p,j,k} \cdot \rho}, \quad \text{(III-24)}$$

$$\eta^i_{p,k} = \sigma^i + \rho^i Y^i_{p,k}, \quad \text{(III-25)}$$

and the worst-case delay $d_{p,k}$ suffered by any priority-$p$ packet at Server $k$ can be bounded by

$$d_{p,k} \le \frac{U_{p,k} - V_{p,k} W_{p,k}}{X_{p,k}}, \tag{III-26}$$

where

$$U_{p,k} = \sum_{q=1}^{p} n_{q,k} \cdot \eta_{q,k}, \tag{III-27}$$

$$V_{p,k} = C_k - \sum_{q=1}^{p} n_{q,k} \cdot \rho, \tag{III-28}$$

$$X_{p,k} = C_k - \sum_{q=1}^{p-1} n_{q,k} \cdot \rho, \tag{III-29}$$

and

$$W_{p,k} = \frac{n_{p,j',k} \cdot \eta_{p,k}}{C_{j',k} - n_{p,j',k} \cdot \rho} \tag{III-30}$$

is defined as the point which satisfies that $s(I) > C_k$ as $I < W_{p,k}$ and $s(I) \le C_k$ as $I \ge W_{p,k}$. In (III-27)-(III-30),

$$n_{q,k}^{i} = \sum_{j=1}^{L_k} n_{q,j,k}^{i}. \tag{III-31}$$

The proof of **Theorem III-2** is given in Appendix A. The delay computation using (III-26) still depends on the number of flows on all input links. In the next subsection, we describe how to remove this dependency.

5.c. Removing the Dependency on the Number of Flows on Each Input Link

As we described earlier, admission control at run time makes sure that the utilization of Server $k$ allocated to flows of Class $i$ with priority $q$ does not exceed $\alpha_{q,k}^{i}$. In other words, the following inequality always holds:

$$n_{q,k}^{i}\rho^{i} \le \alpha_{q,k}^{i}C_{k}. \tag{III-32}$$

The number of flows on each input link is, therefore, subject to the following constraint:

$$n_{q,k}^{i} \le \frac{\alpha_{q,k}^{i}}{\rho^{i}}C_{k}. \tag{III-33}$$

To maximize the right hand side of (III-26), we should maximize $U_{p,k}$ and minimize $V_{p,k}$, $X_{p,k}$, and $W_{p,k}$. Under the constraint of (III-33), these parameters can be bounded for all possible distribution $n_{q,j,k}^{i}$ of numbers of active flows on all input links, as the following theorem shows:

**Theorem III-3.** If the worst-case queuing delay is experienced by any priority- $p$ packet at Server $k$, then,

$$U_{p,k} \le \sum_{q=1}^{p}(\alpha_{q,k}\cdot Z_{q,k})C_{k}, \tag{III-34}$$

$$V_{p,k} \ge (1-\sum_{q=1}^{p}\left\|\alpha_{q,k}\right\|)C_{k}, \tag{III-35}$$

$$X_{p,k} \ge (1-\sum_{q=1}^{p-1}\left\|\alpha_{q,k}\right\|)C_{k}, \tag{III-36}$$

and

$$W_{p,k} \geq \frac{\alpha_{p,k} \cdot Z_{p,k}}{c_k - \left\| \alpha_{p,k} \right\|}. \qquad \text{(III-37)}$$

where $U_{p,k}$, $V_{p,k}$, $X_{p,k}$, $W_{p,k}$ are defined in (III-27)-(III-30), $Z_{p,k}$ is defined in (III-5), *i.e.*, $Z_{q,k}^i = \frac{\sigma^i}{\rho^i} + Y_{q,k}^i$, and $c_k$ is defined in (III-8), i.e., $c_k = \frac{1}{C_k} \sum_{j=1}^{L_k} C_{j,k}$.

The proof of **Theorem III-3** is given in Appendix B.

If we substitute all the bounds in (III-34) into (III-37), then, after some algebraic manipulation, we have

$$d_{p,k} \leq \frac{1}{(1 - \sum_{q=1}^{p-1} \left\| \alpha_{q,k} \right\|)} \left( \sum_{q=1}^{p} \alpha_{q,k} \cdot Z_{q,k} - \frac{(1 - \sum_{q=1}^{p} \left\| \alpha_{q,k} \right\|)}{c_k - \left\| \alpha_{p,k} \right\|} \alpha_{p,k} \cdot Z_{p,k} \right). \qquad \text{(III-38)}$$

(III-3) follows after some definitions of parameters in (III-38). Hence, **Theorem III-1** is proved.

## 6. Priority Assignment

The delay computation formulas described in the previous section allow assignment of priorities to flows independently of their classes. With appropriate priority assignment algorithms in place, network resources can utilized much more effectively.

Ideally, the priority assignment would be done during the admission control for a new flow, where resource usage can be taken into consideration. This would, however, render the admission control procedure significantly more expensive. We, therefore, follow the procedure we used earlier for delay computation and perform the priority assignment off-line, that is, during the system configuration time.

In order to assign priorities to flows off-line, we must classify and aggregate flows using information (in addition to class membership) that is available before run time. For a network with fixed routers, flows can be classified at each server by their class identification, the source and the destination.

In the following, we use class and flow route (in form of source and destination address) information to assign priorities, with all flows in the same class and with the same source and destination having the same priority. This approach has two advantages over more dynamic ones. First, the priority assignment can be done before the admission time and, thus, does not burden the admission control procedure at the time of flow establishment. Second, the static-priority schedulers need no dynamic information at the admission time, as the priority mapping for each packet is fully defined by its class identification and its source and destination identifications. No additional fields in packet headers are needed.

## 6.a. Outline of Algorithms

Mapping with increasing complexity can be used to assign priorities to flows:

- *Algorithm One-to-One:* All flows in a class are assigned the same priority. Flows in different classes are mapped into different priorities. A simple deadline-based mapping can be used to assign priorities to classes, with the earliest deadline getting the highest priority. The advantage of this method is its simplicity. Obviously, this does not take into account more detailed information, such as topology. We use this mapping as the baseline for comparison with other approaches.

- *Algorithm One-to-Many:* Classes may be partitioned into sub-classes for priority assignment purposes, with flows from a class assigned different priorities. Flows in different classes, however, may not share a priority. In Subsection 6.b, we present a version of this algorithm. This algorithm can recognize the different requirements of flows in a class and assign them different priorities, hence, improving the network performance. The algorithm is still relatively simple, but it may use too many priorities since it does not allow priorities to be shared by flows from different classes.
- *Algorithm Many-to-Many:* The priority assignment is not constrained by class membership, and flows from different classes can be assigned the same priority. Given its generality, this mapping can achieve better performance than the previous two algorithms.

### 6.b. Details of Algorithms

We will first focus on *Algorithm One-to-Many*. We will then show that *Algorithms One-to-One* and *Many-to-Many* are a special case and generalization of *Algorithm One-to-Many*, respectively; hence, there will be no need to present the other two algorithms in detail.

The purpose of the static priority assignment algorithm is to generate a *priority assignment table*, which is then used by admission control and, is loaded into routers for scheduling purposes. The priority assignment table (see Table III-1 for an example)

consists of entries of type $\langle class, source, destination, priority \rangle$. The priority assignment then maps from the first three fields in the entry to the *priority* field.

**Table III-1. An Example of Priority Assignment Table**

| Class | Source | Destination | Priority |
|---|---|---|---|
| 1 | Node 2 | Node 3 | 2 |
| 1 | Node 4 | Node 7 | 3 |
| … | … | … | … |
| 3 | Node 6 | Node 1 | 1 |

Algorithm III-1 in Figure III-5 shows our *One-to-Many* priority assignment algorithm. The input of this algorithm is the network server graph, flow routes, characteristic parameters $\langle \sigma^i, \rho^i, D^i \rangle$ for each flow class, and the network bandwidth $\alpha_k^i$ for each class $i$ at Server $k$. We need no knowledge about the exact amount of traffic or the number of flows. The value for the $\alpha_k^i$ can be pre-determined for different class traffic by some policies. The algorithm will return the priority assignment table and bandwidth allocation $\alpha_{p,k}^i$ for each class-$i$ traffic with priority $p$ at Server $k$ or return "FAILURE" as the output.

**Algorithm III-1 Algorithm One-to-Many**

*Input*: Network server graph, flow routes, characteristic parameters $\langle \sigma^i, \rho^i, D^i \rangle$ for each flow class, assigned network bandwidth $\alpha_k^i$, $i = 1,\ldots, M$.

*Output*: Priority assignment table and bandwidth allocation $\alpha_{p,k}^i$, or "FAILURE" if no such bandwidth allocation can be found.

1: initialize the priority assignment table by filling the proper class ID, source ID, and destination ID, and initialize the priority fields to "undefined";

2: for $i$ from $M$ down to 1

2.1: combine all entries of type $\langle i, src, dst, p \rangle$ of Class $i$ into subset $S_i$ and push subset $S_i$ onto Stack $SS$;

3: $p \leftarrow 0$ (initial value of priority)

4: while Stack $SS$ is not empty

4.1: $p \leftarrow p + 1$;

4.2: if $p > P$ (no more priorities available)

4.2.1: return "FAILURE";

4.3: pop a subset $S$ from Stack $SS$; assign $p$ to the priority field of all the entries in $S$; use delay formula (III-3) to update the end-to-end delay of flows represented by entries in $S$;

4.4: if all per-hop laxities of entries in $S \geq 0$ then

4.4.1: continue;

4.5: else if $S$ consists of a single entry

4.5.1: return "FAILURE";

4.6: else call *PartitionSet( S )* and obtain two subsets: $S_x$, $S_y$; push $S_y$ and $S_x$ into Stack $SS$;

5: return the current priority assignment table and $\alpha_{p,k}^i$.

Figure III-5. Algorithm One-to-Many

The algorithm uses a stack to store subsets of entries of which the priority fields are to be assigned. Entries in each subset can potentially be assigned to the same priority. The subsets are ordered in the stack in accordance to their real-time requirements. Given an entry $\langle i, src, dst, p \rangle$ in a subset, assume that the flow route from $src$ to $dst$ is $R$ with length $h(R)$, then we define its per-hop laxity as

$$phl_{i,src,dst,p} = \frac{1}{h(R)}(D_i - d_{p,R}^{e2e}), \tag{III-39}$$

where $D_i$ is the end-to-end deadline requirement and $d_{p,R}^{e2e}$ is the computed worst-case end-to-end delay along route $R$. If the per-hop laxity is less than $0$, packets may miss their deadline. The subset with entries that represent flows with the smallest laxity is at the top of the stack.

After its initialization, the algorithm works iteratively. At each iteration, the algorithm first checks whether enough unused priorities are available. If not, the program stops and declares “FAILURE" (Step 1). Otherwise, a subset is popped from the stack. The algorithm then assigns the best (highest) available priority to the entries in the subset if the deadlines of the flows represented by those entries can be met. However, if some of the deadline tests cannot be passed, Procedure *PartitionSet* (as shown in Algorithm III-2 in Figure III-6) is called to partition the entries in the subset into two subsets based on their per-hop laxity. The idea here is that if we assign a higher priority to entries with little laxity, we may pass the deadline tests for all entries. This is realized by pushing two new subsets into the stack in the proper order and by letting the future iteration deal with the priority assignment. Procedure *PartitionSet* also assigns

bandwidth to traffic with different priorities in the same class. The procedure splits bandwidth according to the ratio of the cardinality of the partitioned subset over the one of the original subset. For example, in this procedure, if $S$ is partitioned into $S_x$ and $S_y$, then $\alpha_{p,k}^i$ will be split into $\frac{|S_x|}{|S|}\alpha_{p,k}^i$ and $\frac{|S_y|}{|S|}\alpha_{p,k}^i$. In our experiments, we use "partition by the half number of entries" (case (i) in Step 2), *i.e.*, $|S_x| = |S_y| = \frac{1}{2}|S|$.

**Algorithm III-2 Procedure PartitionSet(S)**

*Input*: subset $S$

*Output*: subsets $S_x$ and $S_y$, and bandwidth re-allocation $\alpha_{p,k}^i$

1: compute the per-hop laxity for each entry in subset $S$;

2: sort subset $S$ in the increasing order of per-hop laxities;

3: partition $S$ into $S_x$ and $S_y$ (for any entry $s_x \in S_x$ and $s_y \in S_y$, define their corresponding per-hop laxity as $phl(s_x)$ and $phl(s_y)$), such that

(i) $|S_x| = |S_y| = \frac{1}{2}|S|$, or

(ii) $phl(s_x) < \widetilde{phl}(s) \le phl(s_y)$, where $\widetilde{phl}(s)$ is the mean value of per-hop laxities in $S$, or

(iii) $phl(s_x) < 0 \le phl(s_y)$;

4: split $\alpha_{p,k}^i$ into $\frac{|S_x|}{|S|}\alpha_{p,k}^i$ and $\frac{|S_x|}{|S|}\alpha_{p,k}^i$.

Figure III-6. Procedure PartitionSet

The program iterates until either it exhausts all the subsets in the stack, in which case a successful priority assignment has been found and the program returns the assignment table, or it must declare "FAILURE". The latter happens when either the

program runs out of priorities, or it cannot meet the real-time requirements for a single entry in a subset.

Because the size of a subset is split at every iteration step, the worst-case time complexity of the algorithm is in the order of $O(M \log |V|)$ in the number of delay computations. We will show that this algorithm does perform reasonably well in spite of its low time complexity.

*Algorithm One-to-One* is a special case of *Algorithm One-to-Many* presented in Algorithm III-1. For *Algorithm One-to-One*, no subset partition is allowed (otherwise entries in one class will be assigned to different priorities — a violation of the One-to-One principle). Thus, if we modify the code in Algorithm III-1 so that it returns "FAILURE" whenever a failure on a deadline test is found (Step 1), it becomes the code for *Algorithm One-to-One*.

On the other hand, we can generalize *Algorithm One-to-Many* to become *Algorithm Many-to-Many*. Recall that *Algorithm Many-to-Many* allows the priorities to be shared by flows in different classes. Note that sharing a priority is not necessary unless the priorities have been used up. Following this idea, we can modify the code in Algorithm III-1 so that it becomes the code for *Algorithm Many-to-Many*: At Step 1, when it is discovered that all the available priorities have been used up, do not return "FAILURE", but assign the entries with the priority that has just been used. In the case the deadline test fails, assign these entries with a higher priority (until the highest priority has been assigned).

## 7. Performance Evaluation

In this section, we evaluate the performance of the systems that use our new delay analysis techniques and priority assignment algorithms discussed in the previous sections. Recall that we use a utilization-based admission control in our study: As long as the link utilization along the flow route of a flow does not exceed a given bound, the end-to-end deadline of the flow is guaranteed. The value of this bound, therefore, gives a good indication of how many flows can be admitted by the network. We define the *maximum usable utilization* (MUU) to be the summation of the bandwidth portions that can be allocated to real-time traffic in all classes, and use this metric to measure the performance of the systems. For a given network and a given priority assignment algorithm, the value for the MUU is obtained by performing a binary search in conjunction with the priority assignment algorithm discussed in Section 6.

Figure III-7 illustrates the detailed flow chart for MUU Computation. Given a network server graph, flow routes, and characteristic parameters for each class flow (burst, average rate, deadline), we assume that all links have the same utilization, and bandwidth is allocated for different classes traffic by a given ratio pre-determined by policies. For any input of link utilization $u = u_k = \sum_{q=1}^{P} \left\| \vec{\alpha}_{q,k} \right\|$, we calculate the worst-case delay with our delay analysis methods. Then, we can verify whether or not the utilization is safe to make end-to-end delays meet the deadline requirements. Using the binary searching method, we can obtain the maximum usable utilization (MUU) (The

condition of "*stop?*" in Figure III-7 could be "$u_2 - u_1$ is less than the given infinitesimal number or the iteration number is beyond the given limit?").

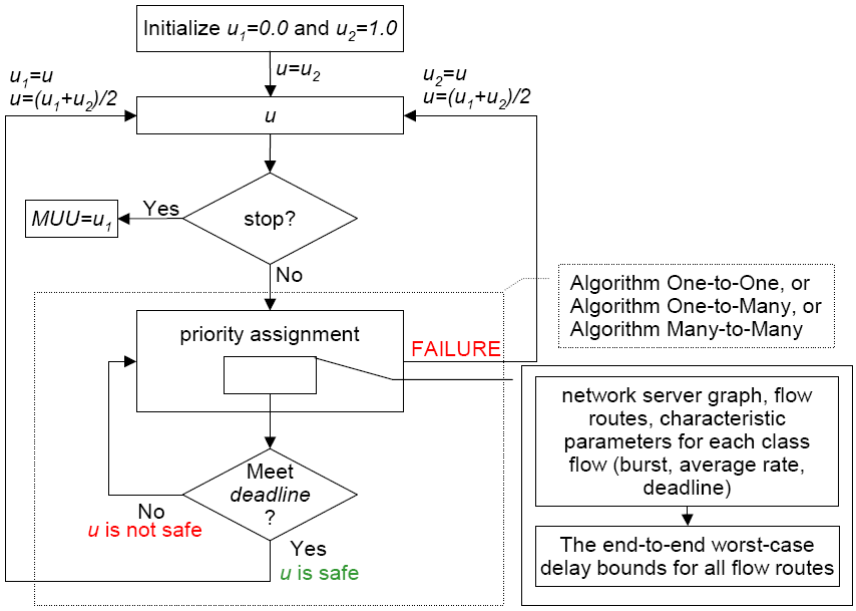



Figure III-7. MUU Computation.

To illustrate the performance of our algorithms for different settings, we describe two experiments. In the first experiment, we use a fixed network topology and compare the performance of the three algorithms presented in Section 6 and measure how the algorithms perform for traffic with varying burstiness. In the second experiment, we measure how the three algorithms behave for networks with different topologies. In the following, we describe the setup for the two experiments and discuss the results.

7.a. Experiment 1

The underlying network topology in this experiment is the classical MCI network topology as shown in Figure III-8. All links in the network have a capacity of 100 Mbps. All link servers in the simulated network use a static-priority scheduler with 8 priorities.

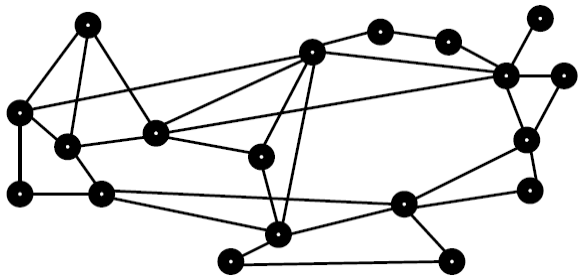


Figure III-8. MCI Network Topology

We assume that there are three classes of traffic: $\langle$ 640 bits, 32,000 bps, 50 ms $\rangle$, $\langle$ 1,280 bits, 64,000 bps, 100 ms $\rangle$, and $\langle$ 1,920 bits, 96,000 bps, 150 ms $\rangle$, where each triple defines $\sigma, \rho$, and the end-to-end deadline requirement for the class. We assume that $\alpha_k^1 : \alpha_k^2 : \alpha_k^3 = 0.05 : 0.10 : 0.20$, which is pre-determined by some policy before hand governing the operation of the network. Any pair of nodes in the simulated networks may request a flow in any class. All the traffic will be routed along shortest paths in terms of the number of hops from source to destination. The results of these simulations are depicted in the first data row of Table III-2. In the subsequent rows of the table, the same experimental results are depicted for higher-burstiness traffic. In each row, the relative burstiness $\frac{\sigma}{\rho}$ is quadrupled.

As expected, Table III-2 shows that the MUU increases significantly with more sophisticated assignment algorithms. The performance improvement of algorithms *One-to-Many* and *Many-to-Many* over *One-to-One* remains constant for traffic with widely different relative burstiness.

**Table III-2. The Comparison of MUU for Different Relative Burstiness**

| $\frac{\sigma}{\rho}$ | MUU | | |
|---|---|---|---|
| | One-to-One | One-to-Many | Many-to-Many |
| 0.02 s | 0.48 | 0.63 | 0.73 |
| 0.08 s | 0.26 | 0.38 | 0.43 |
| 0.32 s | 0.10 | 0.14 | 0.17 |
| 1.28 s | 0.03 | 0.04 | 0.05 |

In Table III-2, we see that the traffic burstiness heavily impacts on the MUU. In fact, for very bursty traffic, the MUU can get quite low. We would like to point out that, even for very bursty traffic, sufficient amounts of bandwidth can still be designated for real-time traffic.

7.b. Experiment 2

In the second experiment, we keep the setup of *Experiment 1*, except we do not vary the relative burstiness of the traffic. Instead, we vary the network topology. We randomly generate network topologies with GT-ITM [81] using the Waxman 2 method described there to generate edges. We classify the generated topologies according to

their size in number of nodes, and their diameter. Performance data are collected based on a sample of 50 networks per number of nodes (overall 550 networks).

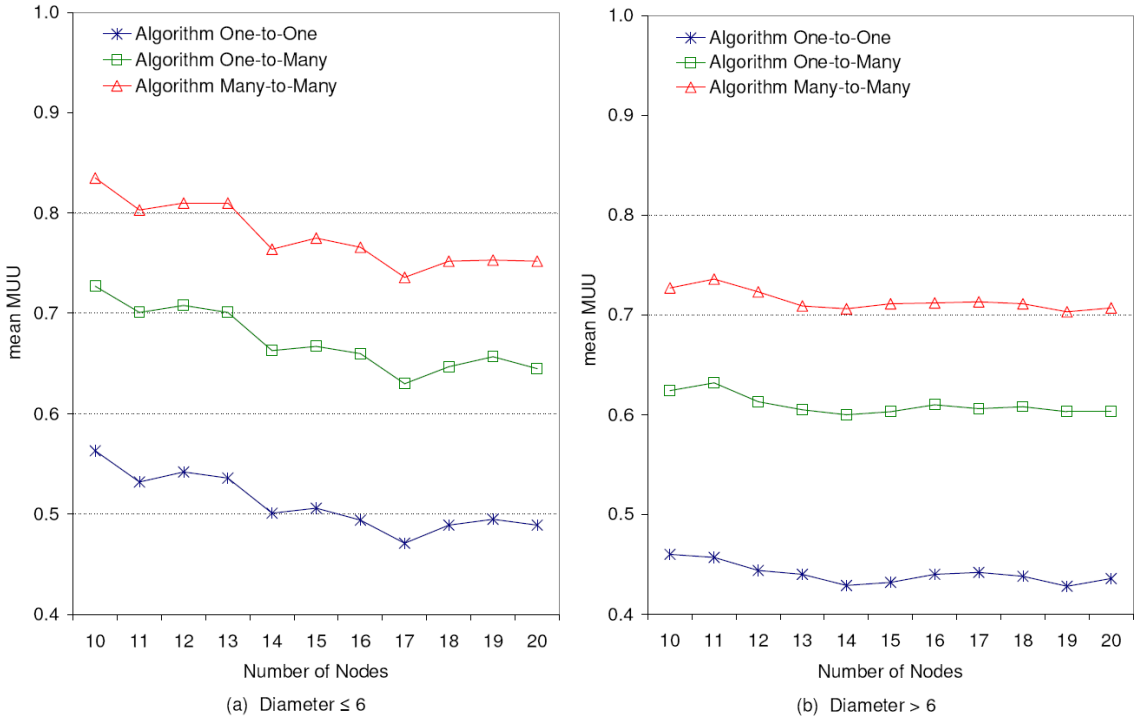



Figure III-9. MUU for Randomly Generated Networks

Figure III-9 displays the mean values for MUU for small networks (diameter of the networks is less than or equal to 6) and for larger networks. We can make the following observations:

- We found that *Algorithm Many-to-Many* can always achieve the highest MUU among the three algorithms, and *Algorithm One-to-Many* can achieve higher mean MUU than *Algorithm One-to-One*, in the networks with the same number of nodes. For example, when the number of nodes is 15, for the case of

$Diameter \leq 6$, the mean MUU of *Algorithm Many-to-Many* is $10.6\%$ higher than that of *Algorithm One-to-Many*, and is $26.7\%$ higher than that of *Algorithm One-to-One*. These observations can be explained by the fact that *Algorithm Many-to-Many* has the highest flexibility in assigning priorities among the three algorithms.

- The diameter of the network has an obvious impact on the performance of all priority assignment algorithms. For example, when the number of nodes is $15$, the MUU of *Algorithm One-to-Many* in the case of $Diameter \leq 6$, is $7.4\%$ higher than that in the case of $Diameter \geq 6$. This is due to the fact that flows in big networks (in the sense of diameter) usually suffer larger end-to-end delay than in small networks.

## 8. Discussion

In this chapter, we use a *worst-case* delay analysis, which follows Cruz's work [66] on worst-case delay analysis. One of our main contributions is the scalable delay analysis technique, which makes offline delay computation possible and allows the adaptation of utilization-based admission control mechanism to be adopted. To achieve this, we have to remove flow-distribution information from the delay analysis, which in turn, makes the worse-case delay bound more loose. Generally speaking, the looser the worse-case delay bound, the lower the achieved maximum usable utilization. By the experimental data, however, the utilization assigned to real-time traffic is still very feasible, *i.e.*, the delay bound is acceptable. For example, as relative burstiness is $0.02$ s,

up to $70\%$ utilization can be assigned to real-time traffic by *Algorithm Many-to-Many*. On the other hand, the residue bandwidth will not be wasted, and can therefore be fully utilized by best-effort traffic (which is assigned the lowest priority).

Considering both run-time delay computation and offline delay computation, we find that there is a trade-off between computational complexity and maximum usable utilization. Run-time delay computation has more run-time admission overhead and, subsequently, larger usable utilization than offline delay computation. In our model, the distribution of flow arrivals is unknown (we only know the characteristic parameters $\left\langle \sigma^i, \rho^i, D^i \right\rangle$ for each class flow). An adaptation method can be adopted to achieve a balance in the trade-off. One possible way is that the utilization can be configured dynamically. During the reconfiguration, all end-to-end delays need to be recomputed and all deadline requirements must be met. More expensive computation may be required but should provide a higher maximum usable utilization.

# CHAPTER IV

# STATISTICAL DELAY GUARANTEES IN WIRED NETWORKS WITH THE DIFFERENTIATED SERVICES MODEL

In this chapter, we extend our work on deterministic delay-guaranteed services and aim at showing how statistical delay-guaranteed services can be provided in a Differentiated Services framework.

## 1. Overview

Although many real-time applications require a hard deadline, i.e., a deterministic delay-guaranteed service, there are still a considerable number of real-time applications that require less rigorous timing constraints, which are often specified in statistical terms. While deterministic delay-guaranteed services provide a very simple model to the application, they tend to heavily overcommit resources because they account for the worst-case scenario, which mostly results in significant portions of network resources (bandwidth etc.) being wasted [82]. Statistical delay-guaranteed services, on the other hand, significantly increase the efficiency of network usage by allowing increased statistical multiplexing of the underlying network resources. This comes at the expense of packets occasionally being dropped or excessively delayed. In this chapter, we will show how such statistical guarantees can be provided in a Differentiated Services framework.

Our approach is based on *rate-variance envelopes* [44] [46], a simple and general traffic characterization. Such envelopes describe the variances of the flow rates as a

function of the interval length. The results on deadline violation probability derived in [44] depend on detailed information about the flow distribution. In this chapter, we will develop flow-distribution-unaware versions of these results, and develop a method that allows us to analyze delays and determine safe utilization bounds without depending on the dynamic status of the flow distribution. This will provide the basis for a scalable admission control for statistical delay guarantees in a Differentiated Services framework.

## 2. Models

We consider the same network model as in the deterministic case. The difference is that a stochastic traffic model should be introduced. We model the traffic arrival for a flow as a stochastic arrival process $f = \{f(t), t \geq 0\}$, where random variable $f(t)$ denotes the incoming traffic amount of the flow passing through a specific point during time interval $[0,t)$. We assume that the stochastic process is stationary and ergodic. The traffic arrivals for any two different flows are stochastically independent at the edge of the network and jitter controllers in the core routers preserve independence throughout the network core. The traffic arrival can be bounded either deterministically or statistically as follows:

**Definition IV-1.** The *function* $F(I)$ is called a deterministic traffic constraint function of the traffic arrival if

$$f(t+I) - f(t) \leq F(I), \tag{IV-1}$$

for any $t>0$ and $I>0$. This is a similar difinition to **Definition III-1**.

**Definition IV-2.** The distribution $B(I)$ forms a statistical traffic envelop of the traffic arrival if, for any $t>0$ and $I>0$,

$$f(t+I) - f(t) \leq_{st} B(I), \tag{IV-2}$$

where $X \leq_{st} Y$ means $P\{X > Z\} \leq P\{Y > Z\}$ for any $Z$.

Since $f$ is stationary, we can define a stochastic arrival traffic rate during a time interval $I$ as

$$R(I) = \frac{f(t+I) - f(t)}{I}. \tag{IV-3}$$

We will be using the rate-variance envelope technique in [46] to provide statistical delay-guaranteed services. The rate-variance envelope $RV(I) = var(R(I))$ describes the variance of the arrival rate for the incoming flow over an interval of length $I$ [46]. It is used as a simple way to capture the second-moment properties of temporally correlated traffic flows.

We consider a simple relationship between priority and class that all flows in the same class are assigned the same priority. We assume that a class-$i$ flow is controlled by a leaky bucket with a bursty size $\sigma_i$ and an average rate $\rho_i$. In the following, we use the notation $G_{i,j}$ to denote the group of flows of Class $i$ from Input Link $j$ of a server (for simplicity, we also ignore the server index). We also use $F_{i,j}(I)$, $B_{i,j}(I)$, and $RV_{i,j}(I)$ to specify the deterministic traffic constraint function, the statistical traffic envelop, and rate-variance envelope applied to the group of flows $G_{i,j}$ respectively.

3. Statistical Delay Guarantees

By *statistical* delay-guaranteed services, we mean that an occasional missed deadline or aborted execution is considered tolerable. A statistical delay guarantee can be defined as a bound on the probability of exceeding a deadline as follows:

$$P\{d > D\} \leq \varepsilon, \tag{IV-4}$$

where the delay $d$ suffered by a packet is a random variable, $D$ is the given deadline, and $\varepsilon$ is the given violation probability, which is generally small. We will see that statistical delay-guaranteed services may significantly increase the efficiency of network usage by allowing increased statistical multiplexing of the underlying resources.

We adopt utilization-based admission control. During system (re-)configuration, a *safe utilization bound* is determined, which is then used for the utilization test at the admission time. As long as the bandwidth usage on the flow route of a new flow does not exceed the safe utilization bound, the performance guarantees of all flows will be achieved. The value for this utilization bound depends on the network topology, the traffic characteristics, and performance requirements of flows.

The challenge of using any utilization-based admission control method is to verify whether a utilization bound is safe or not at the system configuration time. Two critical issues have to be addressed:

- *Statistical Delay Analysis*: The main challenge for statistical delay analysis is how to clearly describe the traffic arrival process. The strong assumptions on the stochastic properties of traffic streams are inherently difficult for the network to enforce or police. Consequently, if a particular application does not

conform to the chosen stochastic model, no guarantees can be made. Moreover, if admitted to the network, such a stream could adversely affect the performance of other applications if it is statistically multiplexed with them. In this paper, we will use an approach previously developed in [44] to conduct the statistical delay analysis, which can cover a wide range of traffic.

- *Utilization-Based Statistical Delay Analysis*: The deadline violation probability derived in [44] depends on the information about flow distribution, i.e., the number of flows at input links and the traffic characteristics (e.g., the average rate and the burst size) of flows. In our case, the delay analysis is done at the system configuration time, when information about flow distribution is not available and only the safe utilization bound is known. Hence, it is necessary to derive a utilization-based formula. We will apply an approach similar to that used in Chapter III for deterministic delay-guaranteed service to solve this.

### 3.a. Statistical Delay Analysis

In statistical delay-guaranteed service, all input traffic conforms to a set of random processes. Suppose these processes are independent. If we know the mean value and the variance of each individual traffic random variable, and the number of flows is large enough, then by the *Central Limit Theorem*, we can approximate the random process of the combined flows. The *Central Limit Theorem* states that the summation of a set of independent random variables converges in distribution to a random variable that has a

*Normal Distribution*.[4] Actually, using rate-variance envelopes, the traffic arrival rate of each individual flow is a random variable, and the mean rate and the rate-variance of each individual flow can be determined using deterministic traffic models. The following theorem can be found in [44]:

**Theorem IV-1.** Consider a static-priority scheduler with $L$ input links and link capacity $C$ such that the traffic with Class $i$ has an random variable delay $d_i$ and an associated deadline $D_i$. Suppose that the group of flows $G_{i,j}$ has mean rate $\phi_{i,j}$ and rate-variance envelope $RV_{i,j}(I)$. With application of a Gaussian approximation over intervals, the deadline violation probability for a random packet with Class $i$ is approximately bounded by

$$\begin{aligned} P\{d_i > D_i\} &\le \max_{I<\beta_i} P\{\frac{1}{C}(\sum_{q=1}^{i-1}\sum_{j=1}^{L} B_{q,j}(I+D_i)+\sum_{j=1}^{L} B_{i,j}(I)) \ge ID_i\} \\ &\le \max_{I<\beta_i} \frac{1}{\sqrt{2\pi}} \exp(-\frac{(C(I+D_i)-\mu_i(I))^2}{2\sigma_i^2(I)}), \end{aligned} \tag{IV-5}$$

where

$$\mu_i(I) = (I+D_i)\sum_{q=1}^{i-1}\sum_{j=1}^{L}\phi_{q,j} + I\sum_{j=1}^{L}\phi_{i,j}, \tag{IV-6}$$

$$\sigma_i^2(I) = (I+D_i)^2\sum_{q=1}^{i-1}\sum_{j=1}^{L} RV_{q,j}(I+D_i) + I^2\sum_{j=1}^{L} RV_{i,j}(I), \tag{IV-7}$$

and

[4] In [44], the author experimentally found the normal distribution approximation to be highly accurate in predicting the performance of a buffered priority multiplexer.

$$\beta_i = \min\{I : \sum_{q=1}^{i}\sum_{j=1}^{L} F_{q,j}(I) \le C \cdot I, I > 0\}. \quad \text{(IV-8)}$$

By this theorem, the deadline violation probability for any random packet can be computed approximately. In the above formula, the final question is how we obtain the values of *mean rate* and *rate-variance envelope*. In [44], two methods are presented for obtaining the rate-variance envelope:

- *Adversarial Mode*: The traffic arrival process conforms to a binomial distribution, where the rate-variance envelope is upper bounded.
- *Non-adversarial Mode*: The traffic arrival process conforms to a weighted uniform distribution, where the rate-variance envelope is approximated but non-worst-case.

Therefore, given the aggregated arrival traffic constraint function $F_{i,j}(t)$ of all $n_{i,j}$ flows, we can specify the *mean rate* and the *rate-variance envelope* as a function of $n_{i,j}$ etc. By **Theorem III-2**, the aggregated arrival traffic constraint function $F_{i,j}(t)$ is given as follows

$$F_{i,j}(I) = \begin{cases} C \cdot I, I \le \tau_{i,j} \\ \mathrm{n}_{i,j}(\sigma_i + \rho_i \cdot I), I > \tau_{i,j} \end{cases} \quad \text{(IV-9)}$$

where

$$\tau_{i,j} = \frac{n_{i,j}\sigma_i}{C - n_{i,j}\rho_i}. \quad \text{(IV-10)}$$

Therefore, we have the following theorem:

**Theorem IV-2.** Given the same condition as **Theorem IV-1**, the mean rate $\phi_{i,j}$ is

$$\phi_{i,j} = n_{i,j}\rho_i, \tag{IV-11}$$

and the rate-variance envelope is upper bounded by

- Adversarial Mode

$$RV_{i,j}(I) \le \frac{1}{I}(n_{i,j})^2 \rho_i \, \sigma_i; \tag{IV-12}$$

- Non-adversarial Mode

$$RV_{i,j}(I) \approx \frac{1}{12I}(n_{i,j})^2 \rho_i \, \sigma_i. \tag{IV-13}$$

The proof of **Theorem IV-2** is given in Appendix E.

At this point, the only undetermined is the number of flows on each link. In the following subsections, we describe how we eliminate the dependency on the number of flows on each link. The result is a delay formula that can be applied without knowledge of the flow distribution.

### 3.b. Utilization-Based Statistical Delay Analysis

As we described earlier, admission control at run time makes sure that the link utilization allocated to each class of flows is not exceeded. The total number $N_i$ of flows of Class $i$ from all input links is therefore subject to the following constraint:

$$N_i = \sum_{j=1}^{L} n_{i,j} \le \frac{\alpha_i}{\rho_i} C, \tag{IV-14}$$

where $\alpha_i$ is the ratio of the link bandwidth allocated to traffic of Class $i$. With this constraint, by (IV-11), (IV-12) and (IV-13), the mean rate and the rate-variance can be upper-bounded. Therefore, the deadline violation probability can be upper-bounded

without relying on the run-time information of flow distribution. This is shown by the following theorem:

**Theorem IV-3.** Consider a static priority scheduler with $L$ input links and link capacity $C$ such that the traffic with Class $i$ has an random variable delay $d_i$ and an associated deadline $D_i$. Suppose that the group of flows $G_{i,j}$ has a statistical envelope $B_{i,j}(I)$. Then the deadline violation probability for a random packet of Class $i$ is bounded by

- Adversarial Mode

$$P\{d_i > D_i\} \le \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2} \min_{I<\beta_i} \xi_i(I)); \tag{IV-15}$$

- Non-adversarial Mode

$$P\{d_i > D_i\} \le \frac{1}{\sqrt{2\pi}} \exp(-6 \min_{I<\beta_i} \xi_i(I)), \tag{IV-16}$$

where

$$\xi_i(I) = \frac{(\eta_i I + \eta_{i-1} D_i)^2}{\zeta_i I + \zeta_{i-1} D_i}, \tag{IV-17}$$

$$\beta_i = \frac{1}{\eta_i} \sum_{q=1}^{i} \alpha_q \frac{\sigma_q}{\rho_q}, \tag{IV-18}$$

and

$$\eta_p = 1 - \sum_{q=1}^{p} \alpha_q, \tag{IV-19}$$

$$\zeta_p = \sum_{q=1}^{p} (\alpha_q)^2 \frac{\sigma_q}{\rho_q}. \qquad \text{(IV-20)}$$

In (IV-19) and (IV-20), the value for $p$ is either $i-1$ or $i$.

This is the main result of this paper. We observe that the formula does not depend on the flow distribution. Hence it can be used for utilization bound verification at configuration time. In the following, we will give the proof. The basic idea is that using inequality (IV-14), the information about the flow distribution will be removed.

*Proof*: Substituting (IV-11), (IV-12) and (IV-13) into (IV-6) and (IV-7), we have

$$C(I + D_i) - \mu_i(I) = I(C - \sum_{q=1}^{i} \sum_{j=1}^{L} n_{q,j}\ \rho_q) + D_i(C - \sum_{q=1}^{i-1} \sum_{j=1}^{L} n_{q,j}\ \rho_q), \qquad \text{(IV-21)}$$

and

- Adversarial Mode

$$\sigma_i^2(I) = I \sum_{q=1}^{i} \sum_{j=1}^{L} (n_{q,j})^2\ \rho_q\ \sigma_q + D_i \sum_{q=1}^{i-1} \sum_{j=1}^{L} (n_{q,j})^2\ \rho_q\ \sigma_q; \qquad \text{(IV-22)}$$

- Non-adversarial Mode

$$\sigma_i^2(I) = \frac{1}{12}( I \sum_{q=1}^{i} \sum_{j=1}^{L} (n_{q,j})^2\ \rho_q\ \sigma_q + D_i \sum_{q=1}^{i-1} \sum_{j=1}^{L} (n_{q,j})^2\ \rho_q\ \sigma_q\ ). \qquad \text{(IV-23)}$$

By (IV-14), we have

$$\sum_{j=1}^{L} n_{q,j}\ \rho_q \le \alpha_q\ C, \qquad \text{(IV-24)}$$

and

$$\sum_{j=1}^{L} (n_{q,j})^2\ \rho_q\ \sigma_q \le (\alpha_q\ C)^2 \frac{\sigma_q}{\rho_q}, \qquad \text{(IV-25)}$$

therfore

- Adversarial Mode

$$\frac{(C(I+D_i)-\mu_i(I))^2}{2\sigma_i^2(I)} \geq \frac{1}{2}\xi_i(I); \tag{IV-26}$$

- Non-adversarial Mode

$$\frac{(C(I+D_i)-\mu_i(I))^2}{2\sigma_i^2(I)} \geq 6\xi_i(I), \tag{IV-27}$$

where $\xi_i(I)$ is defined in (IV-17).□

As an illustrative example, we apply the above theorem in the case of two classes: single real-time class traffic and best effort traffic. Suppose that $\alpha = \alpha_1$, $\sigma = \sigma_1$, $\rho = \rho_1$, $d = d_1$, $D = D_1$, $\xi(I) = \xi_1(I)$, and $\beta = \beta_1$, for real-time class traffic. Simplifying the formula, we can get the following corollary:

**Corollary IV-1.** In the case of a single real-time class, the deadline violation probability for a random real-time class traffic packet is bounded by

- Adversarial Mode

$$P\{d > D\} \leq \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2} \min_{I<\beta} \xi(I)); \tag{IV-28}$$

- Non-adversarial Mode

$$P\{d > D\} \leq \frac{1}{\sqrt{2\pi}} \exp(-6 \min_{I<\beta} \xi(I)), \tag{IV-29}$$

where

$$\xi(I) = \frac{((1-\alpha)I + D)^2}{\alpha^2 \frac{\sigma}{\rho} I}, \tag{IV-30}$$

and

$$\beta = \frac{\alpha}{1-\alpha}\frac{\sigma}{\rho}. \tag{IV-31}$$

Define the right hand side of (IV-28) and (IV-29) as $\varepsilon_1$ and $\varepsilon_2$. We can find that $\xi(I)$ reaches its minimum at $I = I_0 = \frac{D}{1-\alpha}$. By the property of function $\xi(I)$, we find

- if $\beta \geq I_0$, i.e., $\alpha \geq \frac{D}{\frac{\sigma}{\rho}}$, then

$$\varepsilon_1 = \frac{1}{\sqrt{2\pi}}\exp(-2\frac{1-\alpha}{\alpha^2}\frac{D}{\frac{\sigma}{\rho}}), \tag{IV-32}$$

$$\varepsilon_1 = \frac{1}{\sqrt{2\pi}}\exp(-24\frac{1-\alpha}{\alpha^2}\frac{D}{\frac{\sigma}{\rho}}); \tag{IV-33}$$

- if $\beta < I_0$, i.e., $\alpha < \frac{D}{\frac{\sigma}{\rho}}$, then

$$\varepsilon_1 = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}\frac{1-\alpha}{\alpha^3}(\alpha+\frac{D}{\frac{\sigma}{\rho}})^2), \tag{IV-34}$$

$$\varepsilon_1 = \frac{1}{\sqrt{2\pi}}\exp(-6\frac{1-\alpha}{\alpha^3}(\alpha+\frac{D}{\frac{\sigma}{\rho}})^2). \tag{IV-35}$$

By the above formulas, we know that as $\alpha_k$ decrease to below $\frac{D}{\frac{\sigma}{\rho}}$, $\varepsilon_1$ and $\varepsilon_2$ will quickly approach zero.

For the deterministic model, by Theorem 4 in [8], a bandwidth ratio formula can be derived as follows:

$$\alpha = \frac{D}{\frac{\sigma}{\rho}}. \tag{IV-36}$$

Therefore, the bandwidth ratio value in the deterministic model is a critical point to the one in statistical model. Below this value, the deadline violation probability is quite small and quickly approaches zero.

### 3.c. Verification of Utilization Bound

Having derived the utilization-based statistical delay formula, we can verify the utilization bound, and obtain the MUU, the maximum of total utilization bound for all classes. Under the condition that the probabilistic delay guarantee can be met, we can compute the MUU.

For a given deadline violation probability $\varepsilon_i$ and deadline $D_i$ along route $R$, we can split $D_i$ into $\{D_i^k : k \in R\}$, and the delay guarantee is met when

$$P\{d_i^{e2e} > \sum_{k\in R} D_i^k\} \le 1 - \prod_{k\in R}(1 - P\{d_i^k > D_i^k\}) \le \varepsilon_i. \qquad \text{(IV-37)}$$

By **Theorem IV-3**, for the adversarial mode and the non-adversarial mode, substituting (IV-15) and (IV-16) into (IV-37) respectively, we solve the inequalities and get the maximum value of $\alpha_i$, for $i = 1, 2, \cdots, M$. The MUU is $\sum_{i=1}^{M} \alpha_i$.

## 4. Performance Evaluation

In this section, we evaluate the performance of the approaches discussed in the previous sections. We will first define the performance metrics, and then describe the system configuration and present the performance results.

We are interested in two metrics:

- *MUU*: The summation of the bandwidth portions that can be allocated to real-time traffic in all classes. We use this metric to measure the performance of the systems.
- *Admission Probability*: This is the probability that a flow is admitted in a stable system (all packets have bounded delays). The higher the admission probability, the better the network resources are being used.

We assume that the traffic belongs to a single real-time class. We simulate voice traffic, with bursts of 640 bits, an average rate of 32k bit/sec. We assume that the deadline is 5msec. We assume that requests for flow establishment form a Poisson process with rate $\lambda$, and that flow lifetimes are exponentially distributed with an average of 180 seconds. The real system would support best-effort traffic as well, which would not affect the results of this evaluation, and is therefore omitted in these experiments.

The MUU can be computed by (IV-15) and (IV-16) using simple binary search. The admission probability of systems we consider can be analyzed by queuing theory and fixed point method [71].

In the following, we report performance results and make observations. Although we only present a limited number of cases in this dissertation, we find that the conclusions we draw here generally hold for many other cases we have evaluated.

Data on sensitivity of utilization to deadline violation probability are given in Figure IV-1 and Table IV-1. From Figure IV-1 and Table IV-1, we have the following observations:

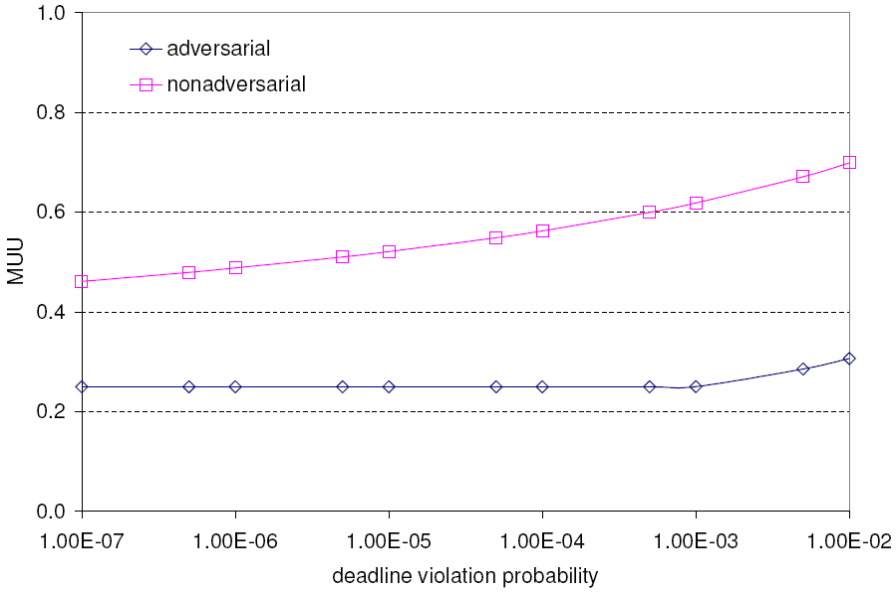



Figure IV-1. Sensitivity of MUU to Delay Violation Probability

**Table IV-1. Sensitivity of MUU to Delay Violation Probability**

| $\varepsilon$ | MUU | | |
|---|---|---|---|
| | deterministic | adversarial | non-adversarial |
| 0 | 0.250 | – | – |
| $10^{-6}$ | – | 0.250 | 0.488 |
| $10^{-4}$ | – | 0.250 | 0.563 |
| $10^{-2}$ | – | 0.307 | 0.699 |

- As expected, the value of the MUU for both adversarial and non-adversarial model increases as the deadline violation probability increases. This means that the higher the deadline violation probability, the higher MUU we can reach for both adversarial and non-adversarial models. A higher deadline violation

probability allows for larger bandwidth allocations and, therefore, higher MUU. Both statistical models can achieve higher or equal value of the MUU than the deterministic model. Since the deterministic model does not allow deadline violations, more resources need to be reserved, which decreases MUU.

- The non-adversarial models achieve much higher MUU than deterministic ones for any deadline violation probability. Non-adversarial models are much closer to real traffic models, and exploit available statistical multiplexing gain more effectively [44].

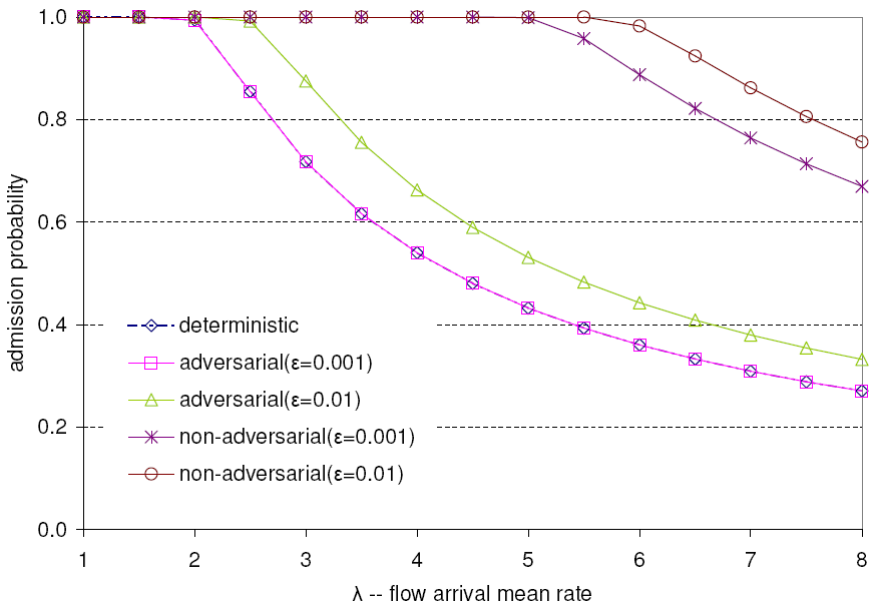



Figure IV-2. Admission Probability Comparison of Deterministic Model and Statistical Model

Data on sensitivity of admission probability is given in Figure IV-2. Note that the curve for deterministic model and the curve for statistical model (adversarial, $\varepsilon = 0.001$) overlaps. From this figure, we make the following observations:

- Admission probability is sensitive to the flow arrival rate $\lambda$ for all models. Admission probability decreases as $\lambda$ increases in all models. The reason is obvious: A large $\lambda$ value implies a large number of flows in the system. Since the bandwidth is limited, some flows are not allowed to enter the network, therefore, the admission probability decreases.
- Different models have different sensitivities to $\lambda$ in terms of admission probability. The statistical models always achieve higher admission probabilities than the deterministic model. Non-adversarial models always achieve higher admission probabilities than adversarial models. This is because of the achievable utilization. The higher this utilization is, the higher the admission probability will be.

# CHAPTER V

# STATISTICAL DELAY GUARANTEES IN WIRELESS NETWORKS*

In this study, we extend our former work on utilization-based delay-guaranteed services in wired networks to wireless networks.

## 1. Overview

The convenience of wireless communications has led to a growing use of wireless networks for both civilian and mission critical applications. Many of these kinds of applications require delay-guaranteed communications. Wireless networks, however, are substantially different from their wired counterparts, and technologies developed for wired networks cannot be directly adopted: In most wired network models for real-time systems, the communication links are assumed to have a fixed capacity over time. This assumption may be invalid in wireless (radio or optical) environments, where link capacities can be temporarily degraded due to fading, attenuation, and path blockage. For example, in a digital cellular radio transmission environment, radio wave reflection, refraction, and scattering, may cause the transmitted signal to reach the receiver by more than one path. This gives rise to the phenomenon known as multipath fading [73]. Also, mobile terminals exhibit time variations in their signal level due to motion [52]. These characteristics of wireless links all result in performance degradation. In order to

---

*Reprinted with permission from "Real-Time Guarantees in Wireless Networks," [72] by S. Wang, R. Nathuji, R. Bettati and W. Zhao, in *Resource Management in Wireless Networking*, M. Cardei, I. Cardei and D.-Z. Du (Eds.), Dordrecht, The Netherlands, Springer, 2005. Copyright 2006 with kind permission of Springer Science and Business Media.

improve the performance of wireless links, error control schemes are used. Common error control methods used in wireless communications include forward error correction (FEC), automatic repeat request (ARQ) and their hybrids [74] [75].

The difficulty of provisioning real-time guarantees in wireless networks stems from the need to explicitly consider both the channel transmission characteristics and the error control mechanisms put in place to alleviate the channel errors.

## 2. Models

In order to provide delay-guaranteed services, one needs both a traffic model, which is the description of the workload carried on links, and an appropriate description of the underlying wireless links. For the traffic model, we adopt the same model used in Chapter IV. We use $F(I)$ for the deterministic traffic constraint function of the traffic arrival, and $B(I)$ as the statistical traffic envelope of the traffic arrival. We will also describe traffic arrivals using the rate-variance envelope [44], which describes the variance of the traffic arrival rate during a time interval. To describe the underlying communication infrastructure in terms of channels and protocols, appropriate models are needed. In the following, we will focus on the wireless link model and describe service capacity with the wireless link model.

### 2.a. Wireless Link Model

We consider a wireless network that consists of a number of wireless links, each of which connects two wireless nodes. This kind of network is used widely in mission-

critical systems ranging from terrestrial-based infrastructures to satellite environments. Figure V-1 shows an example of a wireless network that falls into our network model.

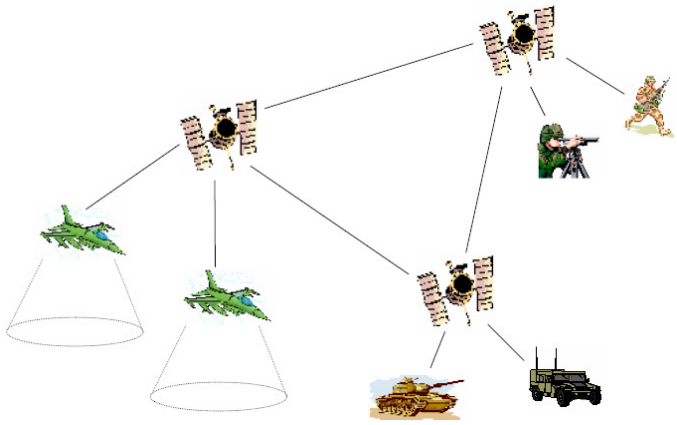


Figure V-1. A Ground-space-ground Wireless Communication System

To guarantee an end-to-end delay, delay characteristics on each wireless link need to be analyzed. Thus, in the rest of this section, we will mostly discuss models related to wireless links in our networks. Underlying wireless links are physical *wireless channels*. For the purpose of delay guarantees, a wireless channel model describes the channel error statistics and its effect on channel capacity. A large number of such models have been described and evaluated in the literature, based on the Rayleigh Fading Channel, or (by adding a line-of-sight component) the Rician Fading Channel [73]. Typical channel error statistics models, such as the binary symmetric channel, are modeled as finite-state Markov models, and can be used to represent time-varying Rician (and other) channels in a variety of settings [76], [77], [78].

In addition to the physical channel, the formulation of a *link model* has to account for error control schemes used at the link layer. In the following, we first consider the framework of the wireless link, and then lay out a more detailed description of our Markov link model. The framework and description will largely follow the approach presented by Krunz and Kim in [53]. We will extend their two-state Markov model to a more general finite-state Markov model.

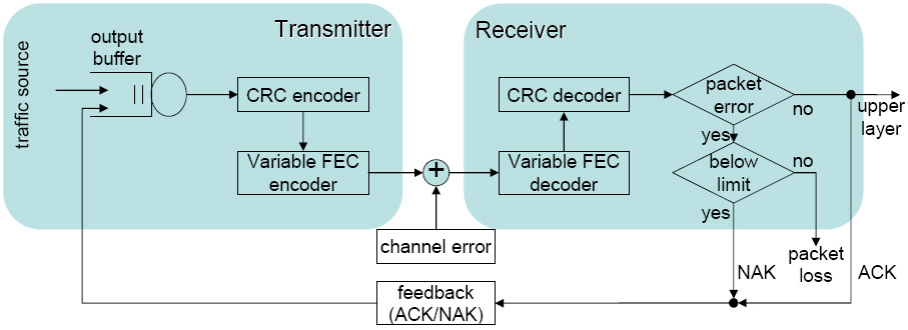



Figure V-2. Wireless Link Framework

We consider a hybrid ARQ/FEC error control scheme (Figure V-2) and assume a *stop-and-wait* (SW) scheme for ARQ: The sender transmits a codeword to the receiver and waits for an acknowledgement. If a positive acknowledgement (ACK) is received, the sender transmits the next codeword. If a negative acknowledgement (NAK) is received, however, the same codeword is retransmitted. NAK's are triggered at the receiver by an error detector, typically based on some form of a cyclic redundancy check.

The FEC capability in the hybrid ARQ/FEC mechanism is characterized by three parameters: the number of bits in a code block ( $n$ ), the number of payload bits ( $k$ ), and

the maximum number of correctable bits in a code block ($r$). Note that $n$ counts the $k$ payload bits and the extra parity bits. Assuming that an FEC code can correct up to $r$ bits and that bit errors in a given channel state are independent, the probability $P_{nc}(p)$ that a packet contains a non-correctable error, given a bit error rate $p$, is given by [53]

$$P_{nc}(p) = \sum_{j=r+1}^{n} \binom{n}{j} p^j (1-p)^{n-j}. \quad \text{(V-1)}$$

To account for the FEC overhead, the actual average service capacity observed at the output of the buffer is $C \cdot \frac{k}{n}$, where $C$ is the maximum capacity for the wireless channel.

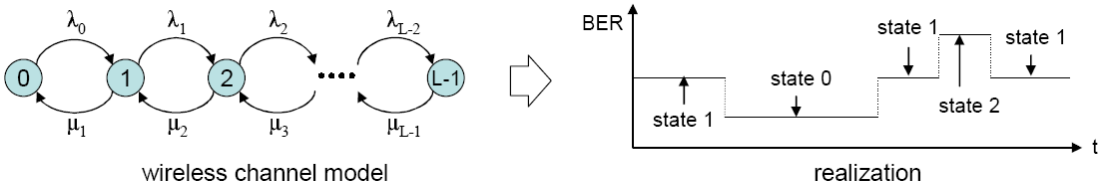



Figure V-3. Fluid Version of Finite-State Markov Model of a Wireless Channel

Although the statistical characteristics of a wireless channel can significantly vary with time, the basic system parameters remain constant over short time intervals. Therefore, we can model the channel to be a quasi-stationary channel. This type of channel can be modeled with finite-state Markov chains [50]. We use a fluid version of a finite-state Markov-Modulated model with $L$ states $(0,1,\ldots,L-1)$ as shown in Figure V-3 [53]. The bit error rates (BER) during State $i$ are given by $p_i$, where we assume $0 \le p_0 < p_1 < \cdots < p_{L-1} \le 1$. The durations in State $i$ before being transitioned to State

$i+1$ and $i-1$ are exponentially distributed with means $\frac{1}{\lambda_i}$ and $\frac{1}{\mu_i}$, respectively. We assume that the transitions only happen between adjacent states.

It is generally difficult to get analytically tractable results that accurately represent the behavior of ARQ and FEC and map the channel model into the respective link model. To solve this, the authors in [53] assume that the packet departure is described by a fluid process with an average constant service capacity that is modulated by the channel state (Figure V-4).

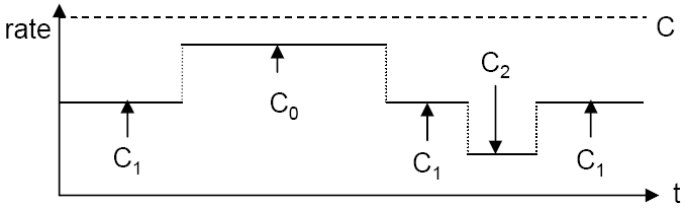



Figure V-4. Approximation Model of a Wireless Link

Each state $i$ then gives rise to a stationary link-layer service capacity $C_i$, which takes packet re-transmissions into account. The total time needed to successfully deliver a packet, conditioned on the channel state, follows a geometric distribution. Let $N_{tr}$ denote the number of retransmissions (including the first transmission) until a packet is successfully received. For the given packet error probability $p_i$ of the channel in State $i$, the expected value for $N_{tr}$ is $E[N_{tr}] = \frac{1}{1-p_i}$ [5]. Thus, $C_i$ can be written as [53]

[5] If we predefine a limit $N_l$ on number of retransmissions, $E[N_{tr}] = \frac{1-p_i^{N_l}}{1-p_i}$ [53].

$$C_i = C \cdot \frac{k}{n} \cdot (1 - P_{nc}(p_i)). \quad \text{(V-2)}$$

As the state transition rates of the channel are not affected by ARQ or FEC, the result is a Markov-modulated model with $L$ state $(0,1,\ldots,L-1)$ with link capacity $C_i$ associated with State $i$.

2.b. Stochastic Service Curve of a Wireless Link

In order to determine the performance guarantees that can be given by a wireless link, we must describe the amount of service that the link can provide. For this, we make use of so-called *service curves*. In the following we show how we derive the service curve from a given link model.

The *stochastic service curve* $S(t) = \int_0^t C(\tau) d\tau$ is defined as the traffic amount that can be served during time interval $[0,t]$ by the wireless channel, where $C(\tau)$ is the capacity at time $\tau$. Correspondingly, we define $S_i(t)$ as the traffic amount that can be served during time interval $[0,t]$ with the system in State $i$ at time $t$, $F_i(t,x)$ and $F_S(t,x)$ as the cumulative probability distribution of $S_i(t)$ and $S(t)$, respectively. We denote $\pi_i$ as the probability that the link is in State $i$ at any time when the system is steady, and we then have

$$F_S(t,x) = \sum_{l=0}^{L-1} \pi_i F_i(t,x). \quad \text{(V-3)}$$

We need to compute $F_i(t,x)$: following a standard fluid approach [79], we proceed by setting up a generating equation for $F_i(t,x)$ at an incremental time $\Delta t$ later in terms of the probabilities at time $t$.

$$\begin{aligned} F_i(t+\Delta t,x) &= (\lambda_{i-1}\Delta t)F_{i-1}(t,x-C_{i-1}\Delta t) \\ &+(1-(\mu_i+\lambda_i)\Delta t)F_i(t,x-C_i\Delta t) \\ &+(\mu_{i+1}\Delta t)F_{i+1}(t,x-C_{i+1}\Delta t), \end{aligned} \tag{V-4}$$

as $i=1,\ldots,L-2$, and

$$F_0(t+\Delta t,x) = ((1-\lambda_1)\Delta t)F_0(t,x-C_0\Delta t)+(\mu_1\Delta t)F_1(t,x-C_1\Delta t), \tag{V-5}$$

$$\begin{aligned} F_{L-1}(t+\Delta t,x) &= (\lambda_{L-2}\Delta t)F_{L-2}(t,x-C_{L-2}\Delta t) \\ &+((1-\mu_{L-1})\Delta t)F_{L-1}(t,x-C_{L-1}\Delta t). \end{aligned} \tag{V-6}$$

Both sides are divided by $\Delta t$ in the above equations. As $\Delta t \to 0$, we have

$$\begin{aligned} &\frac{\partial F_i(t,x)}{\partial t}+C_i\frac{\partial F_i(t,x)}{\partial x} \\ &=\lambda_{i-1}F_{i-1}(t,x)-(\lambda_i+\mu_i)F_i(t,x)+\mu_{i+1}F_{i+1}(t,x), \end{aligned} \tag{V-7}$$

as $i=1,2,\ldots,L-2$, and

$$\frac{\partial F_i(t,x)}{\partial t}+C_i\frac{\partial F_i(t,x)}{\partial x}=\begin{cases} -\lambda_i F_i(t,x)+\mu_{i+1}F_{i+1}(t,x), \mathrm{i}=0 \\ \lambda_{i-1}F_{i-1}(t,x)-\mu_i F_i(t,x), \mathrm{i}=\mathrm{L}-1 \end{cases} \tag{V-8}$$

The initial conditions are $F_i(0,x)=\begin{cases} 0, \mathrm{x}\le 0 \\ 1, \ x>0 \end{cases}$ for $i=0,1,\ldots,L-1$. The partial differential equations can be rewritten in matrix form as follows:

$$\frac{\partial \mathbf{F}}{\partial t}+\mathbf{C}\frac{\partial \mathbf{F}}{\partial x}=\mathbf{Q}\mathbf{F}, \tag{V-9}$$

where $\mathbf{F}=(F_0(t,x),F_1(t,x),\ldots,F_{L-1}(t,x))^{\perp}$, $\mathbf{C}=\mathrm{diag}(C_0,C_1,\ldots,C_{L-1})$ and

$$\mathbf{Q} = \begin{pmatrix} -\lambda_0 & \mu_1 & \cdots & 0 \\ \lambda_0 & -(\lambda_0+\mu_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \mu_{L-1} \\ 0 & 0 & \cdots & -\mu_{L-1} \end{pmatrix}. \tag{V-10}$$

The above linear first-order hyperbolic PDEs can be solved numerically, and the $F_i(t,x)$'s can be computed. Furthermore, if we define $\pi = (\pi_0, \pi_1, \ldots, \pi_{L-1})^{\perp}$, the $\pi_i$'s in (V-3) are given by

$$\pi = \pi\mathbf{Q}, \text{ and } |\pi| = 1. \tag{V-11}$$

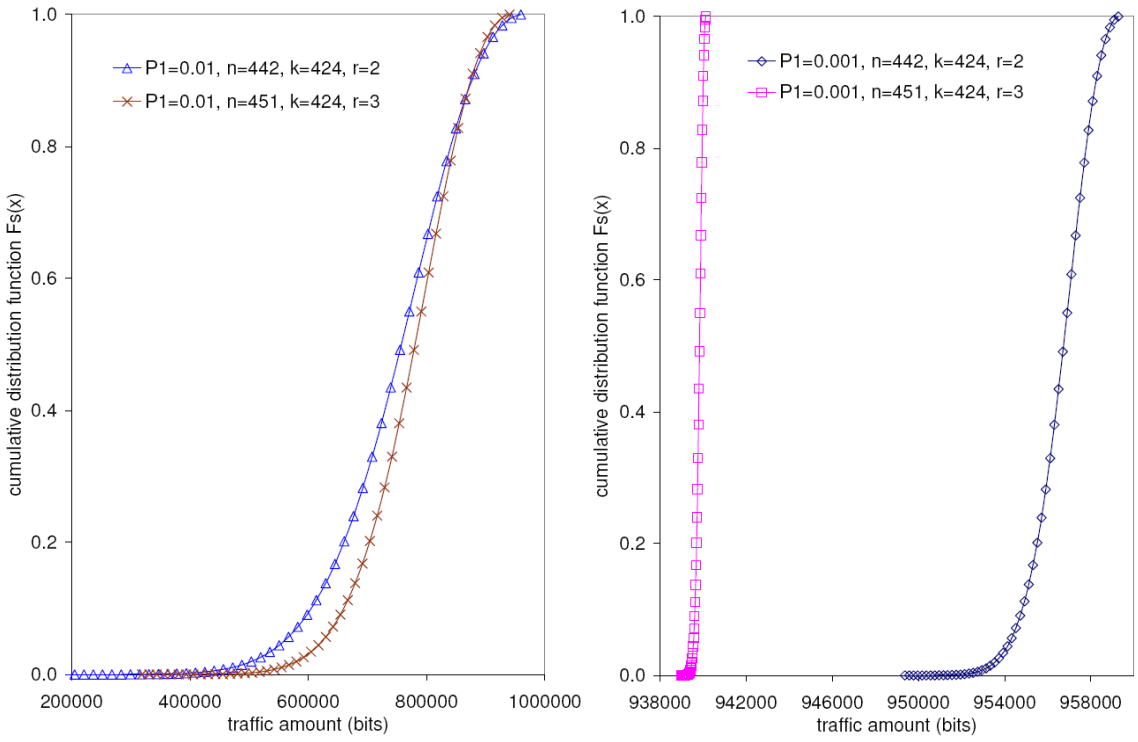



Figure V-5. The Stochastic Service Curve for a Wireless Link

Figure V-5 shows simulated data for the distribution of $S(t)$ for a two-state Markov model, where we specify $C = 2$ Mbps, $\lambda_0 = 10, \lambda_1 = 30, p_0 = 10^{-6}$. We vary the BER $p_1$ and the code parameters $(n, k, r)$. The data illustrates that BER and coding substantially affect the service distribution.

In Section 3, we will illustrate how the service distribution $F_S(t, x)$ can be used to perform statistical delay analysis.

## 3. Statistical Delay Analysis in a Wireless Network

### 3.a. Statistical Delay Analysis

In this subsection, we will perform the delay analysis needed in order to provide end-to-end guarantees. Our analysis will be based on the service description (service curves) introduced in Subsection 2.b and the workload description (traffic arrival) discussed in the beginning of Section 2.

A statistical delay guarantee can be defined as a bound on the probability of exceeding a deadline, i.e., $\Pr\{d > D\} \le \varepsilon$, where the delay $d$ suffered by a packet is a random variable, $D$ is the given deadline, and $\varepsilon$ is the given violation probability, which is generally small.

We consider networks that use static-priority schedulers at the network nodes, as opposed to previous work considering FIFO buffers [53]. For *wired* networks with static priority scheduling, we addressed the issue of how to provide statistical real-time guarantees in Chapter IV, based on Knightly's earlier work in [44]. Define $C$ as the

capacity of a link and $G_i$ as a group of flows that are served by the link at priority $i$. Assume $F_{i,j}(I)$ and $B_{i,j}(I)$ to be the deterministic traffic constraint function and statistical traffic envelope, respectively, for the traffic arrival for the individual flow $j \in G_i$. Then the *deadline violation probability* $\Pr\{d_i \geq D_i\}$ for a random packet with priority $i$ at the output link can be bounded by

$$\Pr\{d_i \geq D_i\} \leq \max_{t<I_i} \Pr\{B^*(I + D_i) \geq C \cdot (I + D_i)\}, \tag{V-12}$$

where $B^*(\cdot)$ is the statistical traffic envelope of aggregated traffic of same and higher priorities:

$$B^*(I + D_i) = \sum_{q=1}^{i-1} \sum_{j \in G_q} B_{q,j}(I + D_i) + \sum_{j \in G_i} B_{i,j}(I), \tag{V-13}$$

and $I_i$ is a bound on the busy interval and is defined as follows:

$$I_i = \min\{I > 0 : \sum_{q=1}^{i} \sum_{j \in G_q} F_{q,j}(I) \geq C \cdot I\}. \tag{V-14}$$

The above formula cannot be applied directly for wireless links however, as their capacities vary over time. Fortunately, as the following observation shows, it is not difficult to integrate stochastic arrivals and a stochastic service curve to compute deadline violation probabilities: Consider a wireless link with a static-priority scheduler and maximum capacity $C$. Let $C(t)$ be the available capacity for traffic as a function of time. Thus $C - C(t)$ is the *unavailable* capacity of link at time $t$. We can equivalently model this system if we define a *virtual traffic arrival* with instantaneous capacity

$C - C(t)$ to a link with constant capacity $C$, by requiring that this virtual traffic is given strictly highest priority during scheduling. Packet delays for real traffic in the original system are identical to delays in this virtual-traffic model. Since the wireless link service capacity is modelled as a stationary process, *S(t)* can be directly applied to *S(I)* in the time interval domain. In particular, if the wireless link has a stochastic service curve $S(I)$ and the system is always steady, then the equivalent virtual traffic on the wireless link has the statistical envelope $B^{'}(I) = C \cdot I - S(I)$. This gives raise to the following theorem:

**Theorem V-1.** Consider a wireless link with a static-priority scheduler and stochastic service curve $S(I)$. Assume $B_{i,j}(I)$ is the statistical traffic envelope for the traffic arrival of the individual flow $j \in G_i$. Then, the deadline violation probability for a random packet with priority $i$ can be bounded by

$$\Pr\{d_i \geq D_i\} \leq \max_{t>0} \Pr\{B^{'}(I + D_i) + B^{*}(I + D_i) \geq C \cdot (I + D_i)\}, \qquad \text{(V-15)}$$

where $B^{'}(I) = C \cdot I - S(I)$, and $B^{*}(t + d_i)$ is defined in (V-13). Here the maximum busy interval is canceled out due to the possibly unconstrained stochastic service curve. The virtual traffic may produce an infinite-length maximum busy interval. So the deadline violation probability may appear to be loose. In our simulation data, we find that the maximum value will be achieved for relatively small values of $t$, therefore, the bound is tight.

We make the following observations about **Theorem V-1**: First, $B^{'}(I+D_i)$ and $B^{*}(I+D_i)$ are independent. Given their distribution functions, the distribution function of the summation $B^{*}(I+D_i)+B^{'}(I+D_i)$ can be obtained by their direct convolution. Second, the distribution of $B^{'}(I+D_i)$ can be directly obtained from $S(I)$, which we in turn derived in Subsection 2.b. Note that (V-15) holds for any $S(I)$, no matter what specific wireless link model is chosen.

The main challenge for statistical delay analysis is how to obtain the distribution function of $B^{'}(I+D_i)$, i.e., how to clearly describe the traffic arrival envelope. It is inherently very difficult for the network to enforce or police the stochastic properties of traffic streams. Consequently, if a particular application does not conform to the chosen stochastic model, no guarantees can be made. Moreover, if admitted to the network, such a non-conforming stream could adversely affect the performance of other applications if it is statistically multiplexed with them. Therefore, we must find a means to describe the non-conforming traffic so that we can perform delay analysis.

We will use the approach previously developed in Chapter IV for the statistical delay analysis. We start by representing the input traffic flows as a set of random processes. Traffic policing ensures that these processes are independent. If we know the mean value and the variance of each individual traffic random variable, and the number of flows is large enough, then by the *Central Limit Theorem* we can approximate the random process of the set of all flows combined. The Central Limit Theorem states that

the summation of a set of independent random variables converges in distribution to a random variable that has a *Normal Distribution*.

We assume that a flow of priority $i$ is controlled by a leaky bucket with burst size $\sigma_i$ and average rate $\rho_i$ at each router. Assume that Flow $j$ in the group of flows $G_i$ has mean rate $\phi_{i,j}$ and rate-variance envelope $RV_{i,j}(I)$. With application of a Gaussian approximation over intervals, $B^*(I+D_i)$ in (V-13) can be approximated by a normal distribution $N(\phi_i(I), RV_i(I))$ [44], where

$$\phi_i(I) = (I+D_i)\sum_{q=1}^{i-1}\sum_{j\in G_q}\phi_{q,j} + I\sum_{j\in G_i}\phi_{i,j}, \tag{V-16}$$

$$RV_i(I) = (I+D_i)^2\sum_{q=1}^{i-1}\sum_{j\in G_q} RV_{q,j}(I+D_i) + I^2\sum_{j\in G_i} RV_{i,j}(I). \tag{V-17}$$

Given the deterministic traffic arrival envelope $F_{i,j}(I) = \sigma_i + \rho_i I$, for any flow $j$ in $G_i$, we can easily obtain mean rate $\phi_{i,j}$ for each individual flow, and an adversarial mode is chosen for obtaining the rate-variance envelope $RV_{i,j}(I)$ [44].[6] We obtain the *mean rate* and the *rate-variance envelope* as follows:

$$\phi_{i,j} = \rho_i, \tag{V-18}$$

$$RV_{i,j}(I) \le \frac{\rho_i\sigma_i}{I}. \tag{V-19}$$

In summary, this leads to the following lemma:

---

[6] In adversarial mode, the traffic arrival process conforms to a binomial distribution, where the rate-variance envelope is upper bounded.

**Lemma V-1.** Define $n_q = |G_q|$, $q = 1, 2, \ldots, i$. With application of a Gaussian approximation over intervals, $B^*(I + D_i)$ can be bounded by a normal distribution $N(\phi_i(I), RV_i(I))$, i.e.,

$$\Pr\{B^*(I + D_i) < x\} \le \Phi\left(\frac{x - \phi_i(I)}{\sqrt{RV_i(I)}}\right), \tag{V-20}$$

where

$$\phi_i(I) = (I + D_i)\sum_{q=1}^{i-1} n_q \rho_q + I n_i \rho_i, \tag{V-21}$$

$$RV_i(I) \le (I + D_i)\sum_{q=1}^{i-1} n_q \rho_q \sigma_q + I n_i \rho_i \sigma_i, \tag{V-22}$$

and

$$\Phi(a) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} \exp\left(-\frac{x^2}{2}\right) dx., \tag{V-23}$$

The distribution function of the summation $B^*(I + D_i) + B'(I + D_i)$ can be obtained by convolution. Define this distribution function as $F_B(I + D_i, x)$. Then, the deadline violation probability can be upper-bounded with utilization as shown in the following theorem:

**Theorem V-2.** Consider a wireless link with a static-priority scheduler and stochastic service curve $S(t)$. Assume the same traffic envelope as in **Theorem V-1**. The deadline violation probability for a random packet with priority $i$ is bounded by

$$\Pr\{d_i \ge D_i\} \le 1 - \min_{t>0} F_B(I + D_i, C \cdot (I + D_i)). \tag{V-24}$$

We have now derived the statistical delay formula for a single wireless link. Based on this result, we obtain the end-to-end deadline violation probability along each flow route as follows: Given the deadline violation probability $\varepsilon_i$ and the end-to-end deadline $D_i$ along route $R$, we can partition $D_i$ into $\{D_i^k : k \in R\}$, and the delay guarantee is met when [45]

$$\Pr\{d_i^{e2e} > \sum_{k\in R} D_i^k\} \le 1 - \prod_{k\in R}(1 - \Pr\{d_i^k > D_i^k\}) \le \varepsilon_i. \qquad \text{(V-25)}$$

This bound on the end-to-end real-time guarantee relies on the information of flow distribution. In next subsection, we will develop utilization-based statistical delay analysis approach, which is independent of such run-time information.

### 3.b. Utilization-Based Statistical Delay Analysis

As opposed to run-time calculations per flow, utilization-based admission control requires off-line delay computations per traffic priority to obtain what we call a *safe utilization bound*. Since flow distribution information is unavailable for off-line calculations, we must obtain a *utilization-based* statistical delay formula, which can be used to compute the safe utilization bound. During run-time, utilization-based admission control checks whether the link utilization allocated to each traffic priority (this allocated utilization should not exceed the safe utilization bound) is not exceeded. The total number $n_i$ of flows of Class $i$ on a link is therefore subject to the following constraint:

$$n_i \le \frac{\alpha_i}{\rho_i} C, \qquad \text{(V-26)}$$

where $\alpha_i$ is the ratio of the link capacity allocated to traffic of priority $i$, and $\rho_i$ is the average rate of priority $i$ traffic. With this constraint, the mean rate and the rate-variance can be upper-bounded as follows:

$$\phi_i(I) = (I + D_i)\sum_{q=1}^{i-1} \alpha_q C + I\alpha_i C, \tag{V-27}$$

$$RV_i(I) = (I + D_i)\sum_{q=1}^{i-1} \alpha_q \sigma_q C + I\alpha_i \sigma_i C. \tag{V-28}$$

Correspondingly, **Lemma V-1**, **Theorem V-1** and Equation (V-25) can be re-formulated using the utilization-based definition for the new $\phi_i(I)$ and $RV_i(t)$ given above.

## 4. Performance Evaluation

In this section, we evaluate the performance of the utilization-based delay guarantee technique. The simulated wireless network could be representative of a ground-space-ground wireless communication system (Figure V-1). We allow any pair of nodes in the network to establish a real-time priority connection (voice in this case). All traffic is routed along the shortest-path route. In our wireless link model, we assume that all links in the network have a maximum capacity of 2 Mbps. Links follow a two-state Markov model as previously defined. In the simulation, we specify the link parameters as follows: $\lambda_0 = 10, \lambda_1 = 30, p_0 = 10^{-6}$, and we vary the bit error rate (BER) $p_1$ for State 1 (BAD state). We also adopt five different Bose-Chaudhuri-Hocquenghem (BCH) [80] coding schemes for FEC. We assume that requests for real-time flow

establishment form a Poisson process, and that flow lifetimes are exponentially distributed with an average of 180 seconds. [7]

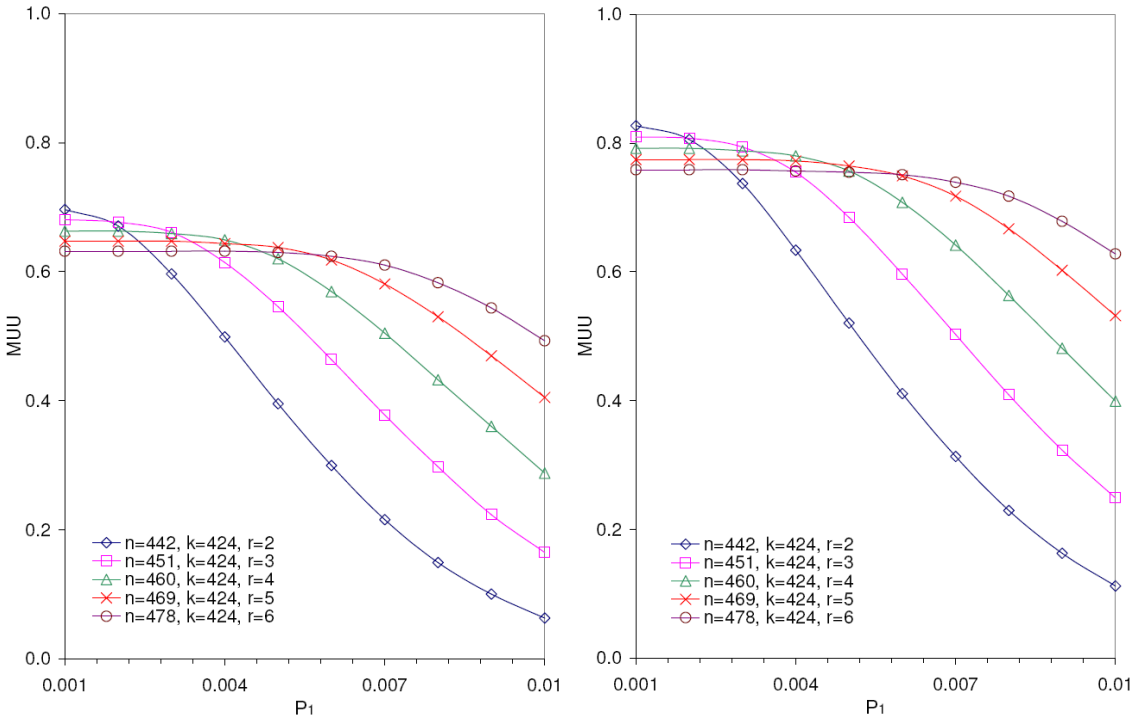
In obtaining our results, we are interested in two metrics: i) *MUU* – The *maximum usable utilization* is the maximum link utilization that can be safely allocated to real-time traffic; ii) *Admission Probability* – This is the probability that a flow can be admitted without violating delay guarantees. Both metrics reflect on the efficient use of network resources.

We find that the conclusions we draw based on the cases described here generally hold for other cases we have evaluated.

### 4.a. MUU Comparison

The underlying network topology in the MUU experiment is the network shown in Figure V-1, where nodes communicate through a space-based reach-back network. We vary the link characteristics by varying the bit error rate (BER) $p_1$ for State 1 (BAD state). We also consider five different BCH coding schemes with increasing level of correctability (i.e., different $(n,k,r)$ [53]. In our traffic model, we assume that all traffic belongs to a single real-time priority. We simulate voice traffic, with bursts $\sigma = 640$ bits, and average rate $\rho = 32000$ bps. We assume that the end-to-end deadline is 15 ms. The end-to-end deadline violation probability is either $10^{-6}$ or $10^{-3}$.

[7] A real system would support best-effort traffic as well. Since this traffic would not affect the results of this evaluation, we omit it from our experiments.




(a) $\varepsilon = 10^{-6}$ (b) $\varepsilon = 10^{-3}$

Figure V-6. MUU Comparison

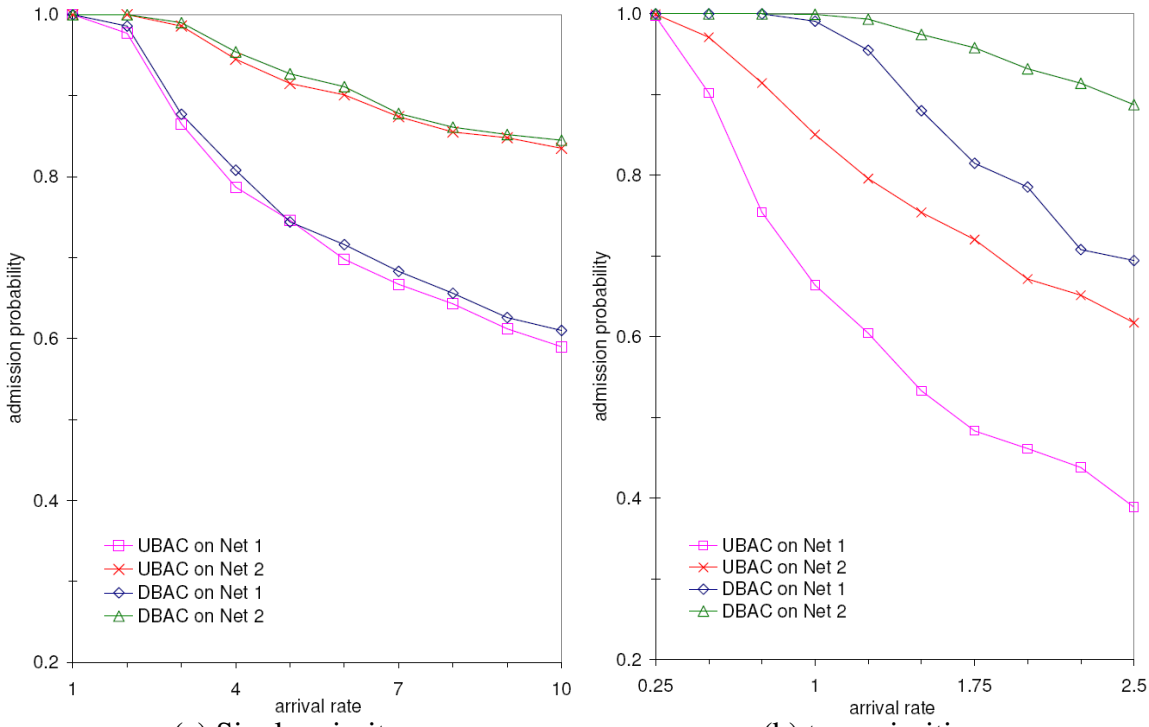
The MUU can be computed by Equation (V-25) that we obtained in the previous section using simple binary search. The results of our MUU experiments are shown in Figure V-6. The following observations can be made from these results: 1). *Sensitivity of MUU to channel coding:* Our results show the performance tradeoff of using various channel codes. Codes that provide greater error correction decrease the amount of actual traffic included in packets. For low error rates, this capability is not worthwhile, as shown in Figure V-6, since error correction is rarely useful, and in fact decreases the overall achievable utilization. 2). *Sensitivity of MUU to BER:* As the BAD-state BER $p_1$ increases from $0.001$ to $0.01$, the MUU decreases for all cases. These results support the intuition that, as the error probability of the network increases, the amount of

capacity that can be supported for real-time traffic should decrease. 3). *Sensitivity of MUU to deadline violation probability:* As expected, the MUU increases when the deadline violation probability is decreased. In other words, allowing higher loss probabilities creates additional available utilization for real-time traffic.

4.b. Admission Probability Comparison

In addition to the topology (Figure V-1) used in the last section (called *Net 1* in this context), we use a random network topology (generated with GT-ITM [81] using the Waxman 2 method) with the same number of total nodes, which we refer to as *Net 2*. We use this randomly generated topology in order to support the fact that our results are not dependent upon a particular topology. We fix bit error rate (BER) $p_1$ for State 1 (BAD state) $p_1 = 0.001$ and choose BCH coding scheme with parameters $(n = 442, k = 424, r = 2)$. The end-to-end deadline violation probability is $10^{-6}$.

We simulate the case when there is only a single real-time priority in the network with same parameters $\sigma, \rho, D$ as the first simulation. We also simulate the case when there are two real-time priorities in the network to see how multiple priorities affect the admission probability. In this case, we choose additional higher-priority traffic as follows: $\sigma = 1280$ bits, $\rho = 64000$ bps, $D = 0.005$ s. The capacity is allocation with ratio $\alpha_{high} : \alpha_{low} = 1:3$.




(a) Single priority (b) two priorities

Figure V-7. Admission Probability Comparison

We measure the admission probability in the system under Delay-Based Admission Control (DBAC)[8] and in our system under Utilization-Based Admission Control (UBAC). As expected, in both single-priority and two-priority cases, the admission probability decreases with increasing flow arrival rate. The substantial conclusion we draw from these results is with regard to the relationship between UBAC and DBAC: It is clear from Figure V-7(a) that in the single-priority case, UBAC is in fact able to provide the same effectiveness with regard to network resource allocation as DBAC. This result is significant because it means that the effectiveness of DBAC can be

[8] The delay computation under DBAC will rely on the delay analysis in Section 3, not the one in Subsection 3.b.

provided with low run-time overhead by using UBAC. Thus, costly run-time delay computations can be removed without sacrificing performance. From Figure V-7(b), we find that DBAC obtains more gains in terms of admission probability than UBAC when there are multiple priorities. This can be attributed to the fact that the pre-allocation of capacity in UBAC disables the capacity sharing between the traffic with different priorities, so that the overall achievable utilization is decreased. Therefore, DBAC achieves much higher admission probabilities than UBAC in the multiple-priority case.

# CHAPTER VI

# DELAY GUARANTEES IN COMPONENT-BASED SYSTEMS

In this chapter, we focus our study on providing delay-guaranteed services in component-based systems, which now serve as an important platform for developing a new generation of computer software. We develop a utilization-based delay guarantee technique to provide efficient and effective delay-guaranteed services while maintaining the important feature of components – reusability.

## 1. Overview

Reusability in component technology is a key factor that contributes to its great success [68]. With component technology, software systems are built by assembling components that have already been developed earlier, with integration in mind. With the software component framework, the non-functional codes, such as security and consistency parts, are automatically generated, and system developers can focus on core business logic parts, without wasting time on common non-functional parts. The reuse of components and developers' focusing on core parts lead to a shortening of software development cycles and savings in software development costs.

Although component-based models deal successfully with functional attributes, they provide little support for delay-guaranteed services. Most real-time extensions use traditional approaches to provide delay-guaranteed services and lack of consideration for reusability of components in terms of both functional and delay-guaranteed services.

In the following, we will develop a utilization-based delay guarantee technique to enable the true reusability of components in terms of both functional and delay-guaranteed services.

## 2. Component-Based Resource Overlays

In component software, a component has three basic characteristic properties [68]: (i) *Isolation* – A component should be deployable independently. The component is an atomic unit of deployment, as it will never be deployed partially. (ii) *Composability* – A component should be composable with other components. It needs to be a self-contained function unit with well-specified interfaces. A third party can access the component through its contractually specified interfaces. (iii) *Opaqueness* – Neither the environment, other components, nor a third party have access to its implementation or other internal details. Figure VI-1 illustrates a component architecture.

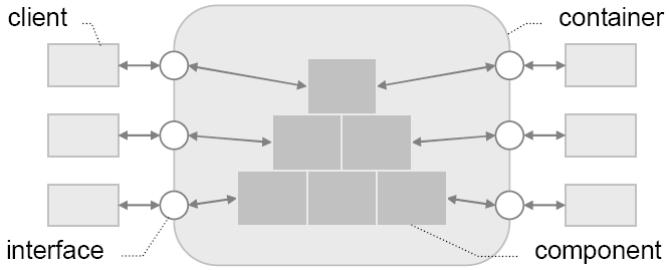



Figure VI-1. A Component Architecture

We extend the component architecture described above to build a *real-time* component architecture. For this, we augment the largely functional interfaces and context dependencies with *contractually specified temporal interfaces* and *explicit time-*

*related context dependencies*. Any such augmentation of the component interface architecture should continue to satisfy the three basic component properties described earlier: (i) The real-time interface architecture should not interfere with the isolation property. Each component should be separated from other components in providing real-time service guarantees. For example, uncontrolled resource conflicts among different components should be avoided. (ii) Composability should be maintained. The real-time service interface should represent the real-time service provided by the component. Applications can access the service through the real-time service interface. (iii) The real-time interface architecture should maintain opaqueness. Applications do not need to know how real-time services are provided by each component. The interfaces should not include information relating to the underlying component implementation, such as methods' worst-case execution time, or any scheduling algorithm used in method execution in components, for example.

We use a very simple contractual interface, which formulates the real-time service provided in terms of the service guarantee (described in the form of a deadline) given a worst-case arrival (described in the form of an arrival constraint function). We first introduce the arrival constraint function.

**Definition VI-1.** If the maximum number of method invocations during any time interval of length $I$ is bounded by $A(I)$, we define $A$ as an arrival constraint function of this sequence of method invocations.

For example, a bursty arrival can be described using a burst size $\sigma$ and average arrival rate $\rho$ as [9] $A(I) = \sigma + \rho \cdot I$. The arrival constraint function $A$ and the deadline $D$ give a contractual definition of the real-time service provided by the component: If the sequence of invocations of methods in Component $e$ has an arrival constraint function below $A$, it is guaranteed that any invocation in this sequence will meet its deadline $D$ at Component $e$. This interface specification clearly meets the isolation, composability, and opaqueness requirements for real-time components.

However, this specification has two shortcomings in practice: First, a component will only provide a single real-time service to applications. Often, different applications may require different levels of timing requirement. Second, each component usually exposes a number of methods and different methods could be invoked at different times, which have different resource consumptions. In order to allow components to provide more flexible service and better utilize the underlying resource usage, we extend the above specification by introducing different service levels and taking into consideration different methods exposed by components. We define *class of service* as the service level for each component. Assuming there are $M$ classes of service, we define class-$i$ real-time service for Component $e$ as $\langle \Theta_{e,i}, A_{e,i}, D_{e,i} \rangle$, where $\Theta_{e,i}$ is a group of methods exposed by Component $e$, $A_{e,i}$ is an arrival constraint function of invocations of methods in $\Theta_{e,i}$, and $D_{e,i}$ is a deadline for any invocation of methods in $\Theta_{e,i}$. In other words, If the sequence of invocations of methods in $\Theta_{e,i}$ has an arrival constraint

[9] Here we use a bound instead of using $\lfloor \cdot \rfloor$ operator

function below $A_{e,i}$, Component $e$ guarantees a worst-case delay bounded by $D_{e,i}$ for any method invocation in this sequence.

**Table VI-1. A Real-Time Service Interface Specification**

| Class $i$ | $\Theta_{e,i}$ | $A_{e,i}(I)$ | $D_{e,i}$ |
|---|---|---|---|
| 1 | $\theta_1, \theta_2$ | $1+2I$ | 0.050 sec |
| 2 | $\theta_1, \theta_2$ | $2+8I$ | 0.250 sec |
| 3 | $\theta_1, \theta_2, \theta_3, \theta_4$ | $3+9I$ | 0.150 sec |
| 4 | $\theta_3, \theta_4$ | $1+4I$ | 0.300 sec |

For example, assume that Component $e$ exposes four methods $\theta_1, \cdots, \theta_4$ and defines four classes, an real-time service interface specification is illustrated in Table VI-1. In this example, $\theta_1$ and $\theta_2$ may represent main methods exposed by the component, while $\theta_3$ and $\theta_4$ are used for management and auditing of the component. Clients that use Class-2 service can access the component at a higher rate than ones that use Class-1 service, but receive less stringent timing guarantees (0.250 sec instead of 0.050 sec). Clients that use Class-3 service have a different view of the component than those that use Class-1 or Class-2 service (they may need to access all methods of the component) and have different real-time requirements. In this example, test suites may need to access all methods exposed by a component, and may need to do this in a timely fashion. Auditing and management applications, on the other hand, may need to access only a small subset of methods, and have only loose time requirements. Note that Class-

1 and Class-2 services expose the same set of methods; that is, $\Theta_{e,1}$ and $\Theta_{e,2}$ are identical. The functional aspect of the service interfaces is therefore identical, while their difference lies entirely in the real-time specification, more specifically in the expected arrival and timing guarantees.

This separation of functional from timing specification allows for a configuration of real-time components into *resource overlays*, which in turn allow for the isolation of applications from the details of the low-level specification and management of the underlying computational resources.

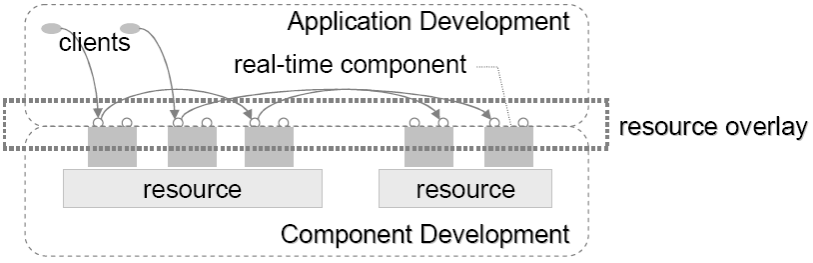



Figure VI-2. A Component-Based Resource Overlay.

Figure VI-2 shows an example of a component-based resource overlay. This figure illustrates how resource overlays separate component development from application development and relieve the application designer of the underlying resource management. In fact, application designers implement and deploy their systems on the resource overlay, which provides well-defined functional abstractions and timing behaviors. The real-time service interface specification in real-time components does not

provide access to their underlying implementation. Any change in the implementation of components will therefore not affect the application design or behavior.

It is up to the component providers to map the nodes of the component-based resource overlay to the underlying available resources. This is typically done independently of the particular application. In the following, we describe in Section 3 how application designers make use of resource overlays to build real-time applications. Given a set of real-time component services, it is the component providers' responsibility to implement the component functionality defined by its set of interfaces. Component providers must ensure that both functional and timing properties are satisfied for each implemented real-time component. We will address this issue in Session 4.

## 3. Building Real-Time Applications

Application designers develop and deploy applications on the resource overlay provided by the set of available real-time components, which in turn expose their real-time service interfaces $\langle \Theta_{e,i}, A_{e,i}, D_{e,i} \rangle$'s to applications and application designers. To build real-time applications, we first introduce the application model.

### 3.a. Application Model

We consider hybrid open/closed systems, where applications include *clients* and *application servers* each of which hosts one or more components. Each invocation from a client triggers execution of one or more methods, either on a single component, or on several components. These components, in turn, can be located on one or across several

application servers. A sequence of client invocations resulting in such a sequence of method executions is called a *task*. In a component-based system, an invocation from a client can pass through several components and we assume all invocations in the same task to execute on the same components in the same order. Tasks in applications can be modeled as a directed acyclic graph which we call *task graph*. Each node is a component and each task forms a *task route*. There could be multiple tasks along each task route.

### 3.b. Service Guarantees with Admission Control

Any task is associated with an arrival descriptor (in form of the source arrival constraint function) and a timing requirement (in form of the end-to-end deadline). To provide real-time service guarantees, application designers have to ensure that every invocation in a task meets the end-to-end deadline requirement. Moreover, the application designer must ensure that the real-time service specified in each real-time component will not be violated. Since the maximum arrival is part of the real-time service of the component, and the client population is not under the control of the application servers, an admission control mechanism has to be in place.

For a new Task $T$, admission control has to address the two parts of the real-time specification of the components (timing guarantee and arrival descriptor) to provide real-time guarantees. *First, what is the worst-case end-to-end delay experienced by any invocation in Task $T$?* In Task $T$, all of its invocations have an end-to-end deadline requirement $D^T$. Assume each invocation in Task $T$ will go through a sequence of components $e_h$ of Class $i_h$, $h = 1, 2, \ldots, H$, and any method that Task $T$ will call in

Component $e$ is in $\Theta_{e,i}$. Recall that the worst-case delay provided by Component $e$ of Class $i$ is $D_{e,i}$. To guarantee the end-to-end deadline for any invocation in Task $T$, admission control has to ensure that the end-to-end delay $d^T$ suffered by any client invocation in Task $T$ should be bounded as:

$$d^T = D_{e_1,i_1} + \cdots + D_{e_H,i_H} \le D^T. \quad \text{(VI-1)}$$

*Second, what is the consumed resource by Task $T$?* Provided that a task has an arrival constraint function $A^{in}(I)$ before arriving at a component, the arrival constraint function will become $A^{out}(I) = A^{in}(I + d)$ just after a worst-case delay $d$ at this component. We define $A^T$ as the source arrival constraint function of Task $T$ (before calling the first component). Then the arrival constraint function of $T$ at Component $e_h$ of Class $i_h$, $h = 1, 2, \ldots, H$, is

$$A^T_{e_h,i_h}(I) = A^T(I + D_{e_1,i_1} + \cdots + D_{e_{h-1},i_{h-1}}). \quad \text{(VI-2)}$$

If $A^T$ is defined with a burst size $\sigma^T$ and an average arrival rate $\rho^T$ as $A^T(I) = \sigma^T + \rho^T I$, then the consumed resource by Task $T$ at Component $e$ of Class $i$ is

$$A^T_{e_h,i_h}(I) = (\sigma^T + \rho^T D_{e_1,i_1} + \cdots + \rho^T D_{e_{h-1},i_{h-1}}) + \rho^T I. \quad \text{(VI-3)}$$

Admission control has to ensure that the real-time service specified at each real-time component along the task route of Task $T$ will not be violated, i.e.,

$$\sum_{T' \in S_{e_h,i_h}} A^{T'}_{e_h,i_h}(I) \le A_{e_h,i_h}(I), \quad \text{(VI-4)}$$

where $S_{e_h,i_h}$ is the set of existing tasks that use class-$i_h$ service of Component $e_h$.

In summary, admission control ensures that sufficient overlay resources are available to meet the requirements of both the new and the existing tasks whenever a new task has been admitted. In other words, both (VI-1) and (VI-4) should remain satisfied for both new and existing tasks if a new task is admitted. This admission control mechanism is simple to implement efficiently. It is utilization-based admission control even though the resource is the virtual one ─ real-time components.

## 4. Building Real-Time Components

Given a set of real-time component services, it is the component providers' responsibility to implement the component functionality so as to satisfy the given set of interfaces $\langle \Theta_{e,i}, A_{e,i}, D_{e,i} \rangle$'s.

### 4.a. Service Implementation

Service implementation issues can be divided into two categories: (i) Inter-component – Recall that each real-time component should be isolated from others in terms of the underlying resource usage to meet the isolation requirement. The underlying resource could be CPU, memory, link bandwidth, or others (here we focus on CPU). We use a *guaranteed-rate scheduler* to ensure temporal isolation of components on the same processor, and we allocate the required amount of the processor utilization to each component. A *total bandwidth server* (TBS) [1] can achieve this; (ii) Intra-component – Each component will provide multiple classes of service. To differentiate among classes

of service within the same component, we use a simple *static-priority scheduler*, and use the class-id as priority level.

The major remaining implementation issue is how to determine the processor utilization that needs to be assigned to each component. We will address this in the following.

Since the component implementation is bound to the underlying hardware platform, the execution of the component's methods can be characterized at component-implementation time. In particular, each exposed method can be associated with its worst-case execution time (WCET) on the specific platform. We aim to compute the worst-case delay suffered by execution of any method in $\Theta_{e,i}$. For this, we denote $C_{e,i}$ as the maximum WCET of all methods in $\Theta_{e,i}$. In conjunction with the arrival constraint function defined as part of the real-time service interface specification, the WCET gives rise to the workload characterization for the method set $\Theta_{e,i}$ on the underlying implementation platform.

**Definition VI-2.** If the cumulative execution time of a sequence of method executions is bounded by $F(I)$ during any time interval with length $I$, we define $F$ as a *workload constraint function* of this sequence of method executions. It is similar to the definition of traffic constraint function defined in **Definition III-1**.

Given the invocation arrival constraint function $A_{e,i}(I) = \sigma_{e,i} + \rho_{e,i} I$ for Component $e$ of Class $i$ and the associated $C_{e,i}$ of $\Theta_{e,i}$, the workload constraint function for Component $e$ of Class $i$ can be expressed as $F_{e,i}(I) = C_{e,i} A_{e,i}(I)$. If we

assume a constant processor utilization $\alpha_e$ to be assigned to real-time Component $e$, we can use a time demand/supply argument to derive the worst-case delay $d_{e,i}$ suffered by any method invocation in Component $e$ of Class $i$ as follows:

$$d_{e,i} \le \max_{I<I_{e,i}}\{\frac{1}{\alpha_e}(\sum_{p<i} F_{e,p}(I+d_{e,i}) + F_{e,i}(I)) - I\}, \tag{VI-5}$$

where $I_{e,i}$ is the maximum busy interval, satisfying

$$I_{e,i} = \min\{I : \frac{1}{\alpha_e}\sum\nolimits_{p\le i} F_{e,p}(I) \le I\}. \tag{VI-6}$$

If $A_{e,i}(I)$ can be defined using burst size $\sigma_{e,i}$ and average arrival rate $\rho_{e,i}$, we can explicitly express $d_{e,i}$ as the following inequality

$$d_{e,i} \le \frac{\sum_{p\le i} C_{e,p}\sigma_{e,p}}{\alpha_e - \sum_{p<i} C_{e,p}\rho_{e,p}} \le D_{e,i}. \tag{VI-7}$$

Therefore, in order to satisfy all classes of service, the allocated processor utilization for components has to be set at least to

$$\alpha_e = \max_{1\le i\le M}\{\frac{1}{D_{e,i}}\sum_{p\le i} C_{e,p}\sigma_{e,p} + \sum\nolimits_{p<i} C_{e,p}\rho_{e,p}\}. \tag{VI-8}$$

When allocating processor utilization to components, component developers should ensure that the overall processor utilization does not exceed the safe utilization level allowed by the specific platform.

4.b. Service Optimization

The functional specifications $\Theta_{e,i}$ of a real-time component can be defined only in the component design, and the timing requirement $D_{e,i}$ is typically defined early on as well. The arrival descriptor $A_{e,i}$, on the other hand, depends on the expected arrival, and requires some understanding of the deployment environment in order to allow efficient resource utilizations. In the following, we describe how component providers can optimally specify arrival descriptor based on the application arrival pattern.

Assume that the task arrival along Task Route $r$ is a Poisson process with average rate $\lambda_r$, the running duration of any task along Task Route $r$ is exponentially distributed with average duration $\frac{1}{\mu_r}$, and any task along Task Route $r$ is associated with real-time source service specification $\langle A_r(I), D_r \rangle$. The average number of tasks along Task Route $r$ is given as $\nu_r = \frac{\lambda_r}{\mu_r}$. Define rejection probability $b_r$ as the probability that a task request for Task Route $r$ is rejected. Then the overall admission probability $AP$ for applications in the system can be expressed as

$$AP = \frac{\sum_{r \in R} \nu_r (1 - b_r)}{\sum_{r \in R} \nu_r}. \qquad \text{(VI-9)}$$

The objective is to find the optimal real-time service specification to maximize the overall admission probability. Then we have the optimization problem as shown in Figure VI-3. It can be summarized as follows:

| | |
|---|---|
| Input: | Task graph $G$ and the set of task routes $R$; $\Theta_{e,i}$ and $D_{e,i}$ for $i = 1, \cdots, M$; $\langle A_r(I), D_r \rangle$, $\lambda_r$ and $\frac{1}{\mu_r}$ for $r \in R$. |
| Output: | $A_{e,i}$'s. |
| Objective: | Maximize the overall task admission probability $AP$. |
| Constraints: | The overall utilization does not exceed the safe utilization level for each application server. |

Figure VI-3. Service Optimization

$$\text{maximize } AP \tag{VI-10}$$

$$\text{subject to } \sum_{e \in \gamma} \alpha_e \le \alpha_\gamma, \gamma \in \Gamma \tag{VI-11}$$

where $e$ is located in Processor $\gamma$ belonging to processor set $\Gamma$ and $\alpha_\gamma$ is the safe utilization bound for Processor $\gamma$. How to solve this optimization problem is addressed in Appendix F.

### 4.c. Service Adaptation

Note that in a component-based system, components compete a limited amount of underlying resources, and the isolation property of components disables the dynamic sharing of the underlying resource among components, which results in an overall resource underutilization. However, due to variations in the application environment, components may not receive constant rate service request from the application at each service level. Based on these observations, it is necessary to use *service adaptation* mechanisms to achieve better resource utilization.

The proposed adaptation scheme, therefore, allows for a load balancing across the component services on the application server. We define the resource residue $\tilde{A}_{e,i}$, i.e., the amount of currently unused resource as

$$\tilde{A}_{e,i}(I) = A_{e,i}(I) - \hat{A}_{e,i}(I), \quad \text{(VI-12)}$$

where $\hat{A}_{e,i}(I) = \sum_{T' \in S_{e,i}} A_{e,i}^{T'}(I)$ is the amount resource currently used by the existing tasks. For any admission request by a task $T$ for component service $\langle e,i \rangle$, we have to borrow resources from some other component service whenever the resources requested by $T$ exceeds the amount of resources currently used, i.e., $\tilde{A}_{e,i} < A_{e,i}^{T}$. We define $A_{e,i}^{*}$ as the optimal service specification obtained by service optimization algorithm and define a threshold $\Delta_{e,i}$ for any real-time component service $\langle e,i \rangle$. If its current assigned resource is no $\Delta_{e,i}$ less than its original optimal assignment, it could be one of candidates whose resources can be borrowed.

The details of this algorithm are shown in Algorithm VI-1 in Figure VI-4. In Step 2, the algorithm identifies the component service with the maximum resource residue, which can be potentially borrowed by other components in the same application server.

Dynamically adapting real-time services of components can improve the statistical multiplexing gain of the underlying resources. We will show this with our evaluation data in Section 7.

**Algorithm VI-1 Service Adaptation()**

*Admission request phase for any Task $T$ :*

1: if $\tilde{A}_{e,i} < A_{e,i}^{T}$ then

1.1: find a component service $\langle \hat{e}, \hat{i} \rangle = \arg\max_{\langle e,i \rangle \in S} \{\tilde{A}_{e,i}\}$, where $S$ is the set of all possible component service satisfying that $A_{e,i}^{*} - A_{e,i} \geq a_{e,i}$ and the overall new utilization will not violate the overall safe utilization bound after resource adaptation;

1.2: update the services of $\langle \hat{e}, \hat{i} \rangle$ and $\langle e, i \rangle$ with their corresponding adapted resource.

*Tear-down phase for any Task $T$ :*

1: undo Step 3 in the above.

Figure VI-4. Service Adaptation

## 5. A Real-Time Component-Based System Architecture

Figure VI-5 displays the system architecture that results from the description above. In this architecture, there is one module – *utilization allocation and scheduling* – for component development, and two modules – *admission control* and *policing* – for application development.

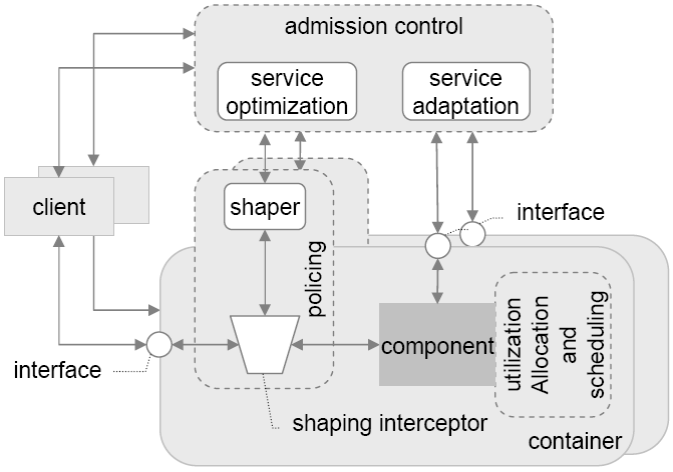



Figure VI-5. System Architecture

Before a real-time Component $e$ is deployed, its real-time service interfaces $\langle \Theta_{e,i}, A_{e,i}, D_{e,i} \rangle$'s must be specified, and will be loaded into the resource table of the *admission control* module once the component is deployed. The *utilization allocation and scheduling* module will preserve the processing utilization assigned to each component with the resource reservation mechanism and schedule the execution of methods at each component with the scheduling algorithm.

Admission control is performed at task level: When the client wants to start a new task, it first sends an admission request to the *admission control* module. The admission control module will make a decision for this admission request based on the policy in *admission control mechanism* and the profile including in the admission request. If the admission request is admitted, an acknowledgement message will be sent back to the client, which includes a task ID. At the same time, shaper instances at the corresponding components for this task will be created. During the tear-down process of this task, the

task will be removed from the existing-task table. Information about the admitted task will be maintained in the *existing-task table* in the *admission control* module. The modules for *Service Optimization* and *Service Adaptation* are implemented as sub-modules in *admission control* module.

When a task is successfully admitted, the system will protect its resources against sources of invocations that exceed their share of invocation arrivals. This is done by appropriately policing the invocations by the *policing module*. One of the most well-known policing mechanisms in the literature is known as *leaky bucket*, in which the arrival constraint function is defined by a burst size and an average arrival rate. Once policed, the invocations are passed on to *utilization allocation and scheduling* module for execution of their corresponding method on the processor.

## 6. Implementation of a Real-Time Component-Based System

We used Enterprise JavaBeans (EJB) [56] as the underlying framework of a resource overlay infrastructure based on real-time components. In the following, we first introduce the background information, such as EJB and its implementation JBoss, and then address the implementation of our system in details.

### 6.a. EJB and JBoss Application Server

EJB technology is a server-side component architecture that simplifies the development and deployment of multi-tier, distributed, scalable, Java enterprise applications. Enterprise beans (beans, to be concise) are server-side components in EJB, which represents a business concept. There are three basic types of beans: (i) *entity*

*beans*, which represent data in a database, (ii) *session beans*, which represent processes or act as agents performing tasks, and (iii) *message-driven beans*, which are asynchronous message consumers.

JBoss [85] application server is a popular, open-source EJB application server. It provides the basic EJB containers as well as EJB services such as database access(JDBC), transactions (JTA/JTS), messaging (JMS), naming (JNDI) and management support (JMX).

As the foundation for the JBoss infrastructure, JMX [86] provides a common server spine that allows the user to integrate modules, containers, and plug-ins. Service components are declared as Managed Bean (MBean), which are then loaded into JBoss and may subsequently be administered using JMX. MBeans are managed resources and Java objects that follow certain conventions to expose their management interfaces to remote management applications. Remote management applications can access MBeans through JMX agent services. Each MBean is given a unique *object name* and registered to MBean Server at initial time. MBean Server provides a registry service for MBeans. The JBoss EJB server and EJB container are completely implemented using component-based plug-ins onto JMX. When an EJB is deployed into JBoss, a container MBean is created to manage the EJB [87], [88]. In our real-time component-based system, we will build the admission control module as an MBean.

In JBoss, the dynamic proxy approach is used for the server to generate container classes and for the generated container to generate home and remote interfaces of the EJB at run time. Whenever a method invocation is issued on the client-side proxy, the

invocation handler creates a special Invocation object, which will reify the method invocation. After traversing a chain of *client-side interceptors*, the Invocation object is sent by an *invoker proxy* to an *invoker MBean* at the server side, where it is routed through the *container MBean* associated with the target EJB. Each *Invocation* object includes information about object name, method, and arguments for target EJB. Application developers can place customized interceptors in the *interceptor stack* traversed by the *Invocation* object. This interceptor stack mechanism allows developers to add additional services to the called target [87]. We use this interceptor mechanism to build the policing module in our real-time component-based system.

### 6.b. Real-time Infrastructure

The implementation of our real-time component-based system is based on JBoss 3.2.1. We use TimeSys Linux RT 3.1 as the underlying real-time operating system [90]. To complete the platform, we use the RTSJ Reference Implementation (RTSJ-RI) from TimeSys [91] as Java VM. Since TimeSys RTSJ-RI (just as other foreseeable RTSJ implementations) provides only a limited set of Java classes, and JBoss is intended for building enterprise systems, we disabled some advanced features in JBoss while at the same time adding RTSJ compatible Java class libraries. In particular, RTSJ-RI does not support real-time capabilities for Java Remote Method Invocation (RMI) [89] on which remote invocation is based. As a result, we appropriately extended RMI to make it real-time capable. For this, we eliminated sources for priority inversion, such as for situations in which the listening thread used to assign incorrect priorities to incoming requests. As a result, this combination of a real-time OS, a real-time capable Java, and Real-Time

RMI in conjunction with an appropriately trimmed JBoss framework results in a powerful basis for a real-time component-based system.

## 6.c. Implementation of Real-Time Component-Based System

The core of the real-time component-based system as described in the previous sections is the Real-Time Specification, the admission control, the policer, and the thread scheduler.

*Real-Time Service Specification*: Each real-time component will expose its real-time service interface. The real-time service is defined as a number of *ClassOfService* objects. Class *ClassOfService* is defined in Figure VI-6, where Class ArrivalFunction defines the arrival constraint function for the corresponding class of service.

```
class ClassOfService {
    int classID;
    Method[] groupOfMethods;
    ArrivalFunction arrival;
    long deadline;
    ... // methods not shown
}
```

Figure VI-6. Real-Time Service

*Admission Control Module*: The admission control module is implemented as an MBean. This MBean realizes the admission control mechanism and the decision making

procedure. The resource table and the existing-task table required by admission control are implemented as entity beans. Service optimization and service adaptation are implemented as sub-modules.

*Policing Module*: After a task request is admitted, a task ID is generated and forwarded to the client. At the same time, a shaper instance (implemented as entity beans) is created. Any invocation of this task will add the task ID to the Invocation object at the AdmissionInterceptor on the client side. At the ShapingInterceptor on the server side, the task ID is retrieved, and is used, together with the name of the EJB, as a key to match its corresponding shaper instance.

*Utilization Allocation and Scheduling Module*: As an example of a guaranteed-rate scheduler to provide temporal isolation, a Total Bandwidth Server (TBS) [1] is implemented to allocate the underlying CPU utilization to each real-time component. The input parameters for TBS are WCETs of methods in the component and the allocated CPU utilization. Recall that when a task is admitted, a corresponding shaper instance is initiated. The shaper instance also includes the priority assigned to any run-time method execution in the task. Once an invocation enters ShapingInterceptor, its corresponding shaper instance can be found in the same way as done in the *policing module*. Then the priority value can be retrieved, and the priority for the worker thread is set.

## 7. Performance Evaluation

Recall that one of the basic features in our designed systems is the reusability of real-time components. Reusability, however, is difficult to evaluate quantitatively. In the experiments described below, we focus on the performance of our system in terms of the admission probability for each task request and the latency overhead introduced in our system. In our experiments, we assume that the CPU on the EJB server is the bottleneck and that the other resources, such as memory, disk and network bandwidth are never limiting.

### 7.a. Admission Probability versus Task Arrival Rate

In this experiment, we choose two Pentium 4 machines with 2.53 GHz CPU and 1 GB memory as application servers. These two machines are in the same subnet together with another machine chosen to host as the clients. The network bandwidth for all connections is 100 Mbps. The application servers are installed with real-time component-based systems software.

We deployed three real-time components $\{e_1, e_2, e_3\}$ – real-time session beans – in the first real-time application server and $e_4$ in the other real-time application server. Component $e_i$ will expose one method $\theta_{e_k}$ and define a single class of service (therefore we can ignore the class index), and its real-time service interface is $\left\langle \Theta_{e_k}, A_{e_k}, D_{e_k} \right\rangle$, where $\Theta_{e_k} = \{\theta_{e_k}\}$, and $D_{e_k} = 1.000$ sec, for $k = 1,\ldots,4$. Methods are associated with WCET $C_{e_1} = 0.050$ sec, $C_{e_2} = 0.080$ sec, $C_{e_3} = 0.050$ sec, and $C_{e_4} = 0.030$ sec,

respectively. The safe utilization for each processor at each application server is $90\%$. The arrival constraint function $A_{e_k}(I)$ will be optimally specified and CPU utilization $\alpha_{e_k}$ will be determined with our service optimization algorithm. The runtime method execution will be assigned a single real-time priority. There are three task routes as shown in Figure VI-7.

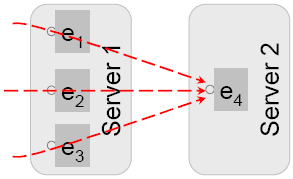



Figure VI-7. The Experiment Testbed

In this experiment, we choose three different system configurations in the application server: A system with no component isolation and enabled admission control (NCI), our system with component isolation and enabled admission control under service optimization (CI-OPT), and our system with component isolation and enabled admission control under service adaptation (CI-SA). In CI-SA, we choose $\Delta_{e,i} = 2A^T_{e,i}$. Emulated applications consist of task generators, and client-server interactions. Clients will send a sequence of periodic tasks. In each task, the period for invocation arrival is $2$ sec and each invocation has a $2$ sec deadline requirement. Each task has a exponentially-distributed life time and includes $6$ invocations in a life time in average. The task arrival is a Poisson process and each task will choose a task route uniformly randomly. We vary the overall task arrival rate $\lambda$ from $0.1$ per sec to $1.5$ per sec.

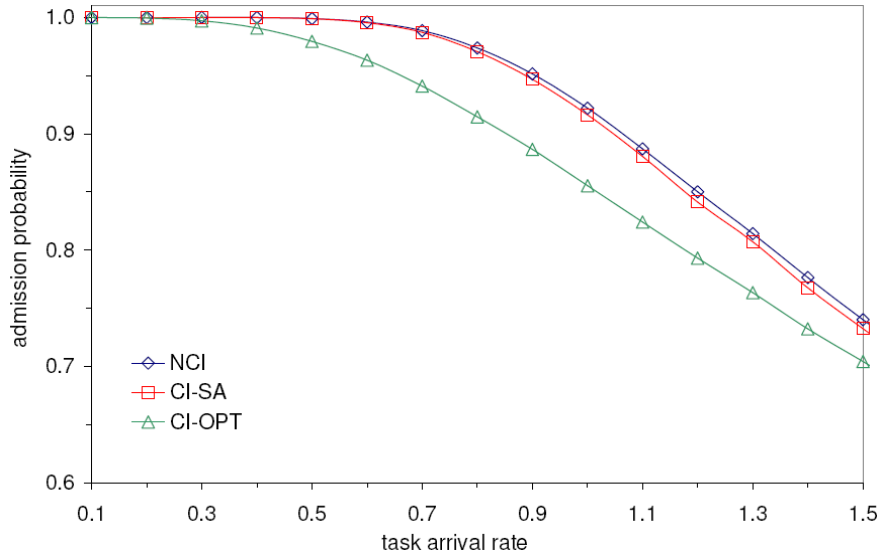



Figure VI-8. Comparison of Admission Probability vs. Task Arrival Rate

Figure VI-8 shows admission probabilities for all task admission requests from clients. As expected, as the task arrival rate increases, admission probability decreases in the systems with enabled admission control for all system configurations. The data show that CI-OPT has a lower admission probability than NCI with only a maximal difference $6.7\%$ as $\lambda = 1.0$. With our introduced adaptation algorithm, the admission probability in CI-SA can be improved close to the one in NCI as shown in the Figure.

7.b. Admission Probability versus Number of Components

In this experiment, we deploy $n$ real-time components in the first application server and none in the second application server. Each component exposes one method with WCET $C_{e_k} = 0.050$ sec, $k = 1, 2, \cdots, n$. In each periodic task, the period for invocation arrival is $1$ sec and each invocation has a deadline $1$ sec requirement. We fix the overall task arrival rate as $2.0$ per sec. The other configurations are the same as the

experiment above. We vary $n$ from 2 to 10 and measure the admission probability for each $n$.

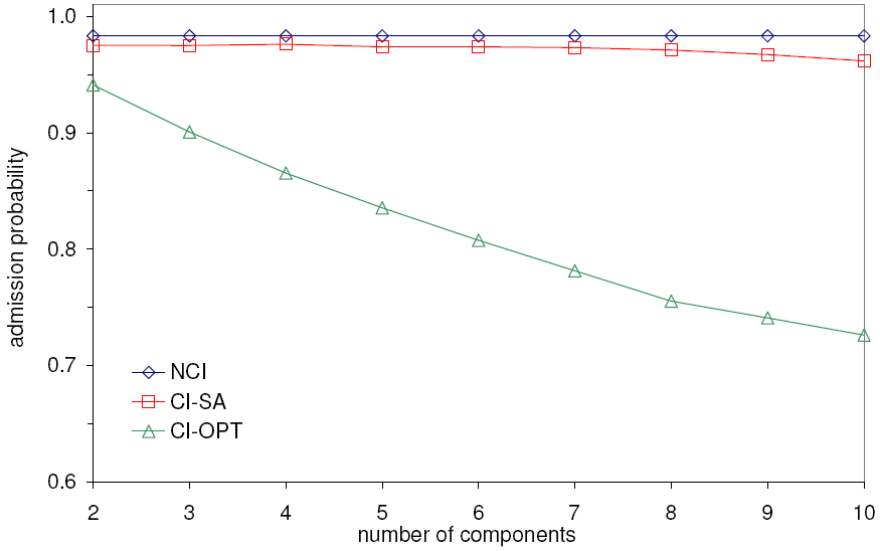



Figure VI-9. Comparison of Admission Probability vs. Number of Components

Figure VI-9 shows that the admission probability in NCI keeps constant and the one in CI-OPT will decrease as the number of components increases. With the adaptation algorithm, the admission probability in CI-SA can still be improved close to the one in NCI as shown in the Figure.

### 7.c. Admission Control Latency

In the first experiment, we also collect the data about the latency of admission decision conducted by admission control. In Figure VI-10, the data shows that the average latency will increase slowly as the task arrival rate increases.

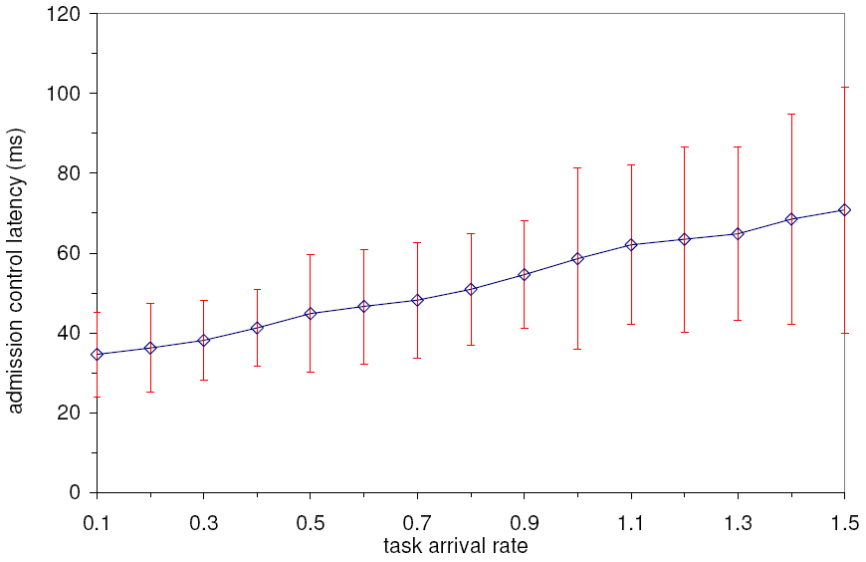



Figure VI-10. Admission Control Latency

Recall that the admission control module is implemented as an MBean and the involved resource table and existing-task table are implemented as entity beans. This implementation results in the admission control latency up to 0.035 sec in average as $\lambda = 0.1$. Since the queueing for the bursty task arrival may introduce an extra delay, the latency will increase slowly as the task arrival rate increases. The admission control latency only increases 2.02 times as the task arrival rate increases 15 times (from 0.1 to 1.5). Our admission control mechanism is relatively scalable.

# CHAPTER VII

# SUMMARY

In this dissertation, we have presented our methodology of developing utilization-based delay guarantee techniques and applying them in systems using the utilization-based admission control mechanism.

We first considered wired networks with the Differentiated Services model and proposed utilization-based delay guarantee techniques to make Differentiated Services systems support both deterministic and statistical delay-guaranteed services. We then extended our work to wireless networks, in which it is more challenging to support delay-guaranteed services due to the variable link capacity. Finally, we study ways to provide delay-guaranteed services in component-based systems while maintaining the reusability feature of components.

We showed that with our utilization-based delay guarantee techniques, the admission control mechanism can be made both efficient and effective in providing delay-guaranteed services.

# REFERENCES

[1] J. W. Liu, *Real-Time Systems*. Upper Saddle River, NJ: Prentice Hall, 2000.

[2] D. Collins, *Carrier Grade Voice over IP*. New York, NY: McGraw-Hill, 2003.

[3] R. Nitzberg, *Radar Signal Processing and Adaptive Systems*. Boston, MA: Artech House, 1999.

[4] A. Dailianas and A. Bovopoulis, "Real-time admission control algorithms with delay and loss guarantees in ATM networks," in *Proc. IEEE INFOCOM*, Toronto, Canada, June 1994, pp. 1065–1072.

[5] V. Firoiu, J. Kurose, and D. Towsley, "Efficient admission control for EDF schedulers," in *Proc. IEEE INFOCOM*, Kobe, Japan, April 1997, pp. 310–317.

[6] J. Liebeherr, D. Wrege, and D. Ferrari, "Exact admission control in networks with bounded delay services," *IEEE/ACM Trans. Networking*, vol. 4, pp. 885–901, Dec. 1996.

[7] B.-K. Choi, D. Xuan, C. Li, R. Bettati, and W. Zhao, "Scalable QoS guaranteed communication services for real-time applications," in *Proc. IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, Taipei, Taiwan, April 2000, pp. 180–187.

[8] D. Xuan, C. Li, R. Bettati, J. Chen, and W. Zhao, "Utilization-based admission control for real-time applications," in *Proc. Int. Conf. on Parallel Processing (ICPP)*, Toronto, Ont., Canada, Aug. 2000, pp. 251–260.

[9] B.-K. Choi and R. Bettati, "Endpoint admission control: network based approach," in *Proc. IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, Phoenix, AZ, April 2001, pp. 227–235.

[10] B.-K. Choi , D. Xuan, R. Bettati, W. Zhao, and C. Li, "Utilization-based admission control for scalable real-time communications," *J. Real-Time Systems*, vol. 24, pp. 171–202, Mar. 2003.

[11] W. Zhao, “A heuristic approach to scheduling hard real-time tasks with resource requirements in distributed systems,” Ph.D. dissertation, Dept. of Comput. Sci., Univ. of Massachusetts, Amherst, MA, Feb. 1986.

[12] R. Bettati, "End-to-end scheduling to meet deadlines in distributed systems," Ph.D. dissertation, Dept. of Comput. Sci., Univ. of Illinois, Urbana-Champaign, IL, Aug. 1994.

[13] C. Li, "Analysis of network traffic and its applications," Ph.D. dissertation, Dept. of Comput. Sci., Texas A&M Univ., Dec. 1999.

[14] B. K. Choi, "Resource management for scalable quality of service," Ph.D. dissertation, Dept. of Comput. Sci., Texas A&M Univ., Dec. 2002.

[15] J. Wu, “General schedulability bound analysis and its applications in real-time systems,” Ph.D. dissertation, Dept. of Comput. Sci., Texas A&M Univ., May 2006.

[16] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” *J. ACM*, vol. 20, pp. 46–61, Jan. 1973.

[17] C.-C. Han, and H.-Y. Tyan, “A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithm,” in *Proc. IEEE Real-Time Systems Symp. (RTSS)*, San Francisco, CA, Dec. 1997, pp. 36–45.

[18] J. P. Lehoczky and L. Sha, “Performance of real-time bus scheduling algorithms,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 14, pp. 44–53, May 1986.

[19] A. K. Mok and D. Chen, “A general model for real-time tasks,” Dept. Comput. Sci., The Univ. of Texas, Austin, TX, Tech. Rep. TR-96-24, Oct. 1996.

[20] A. K. Mok and D. Chen, “A multiframe model for real-time tasks,” *IEEE Trans. Software Engineering*, vol. 23, pp. 635–645, Oct. 1997.

[21] D.-T. Peng and K.G. Shin, “A new performance measure for scheduling independent real-time tasks,” *J. Parallel Distributing Computing*, vol. 19, pp. 11–16, Sep. 1993.

[22] J. Wu, J.-C. Liu, and W. Zhao, “On schedulability bounds of static priority schedulers,” in *Proc. IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)*, San Francisco, CA, Mar. 2005, pp. 11–16.

[23] B. Andersson, "Static-priority scheduling on multiprocessors," Ph.D. dissertation, Dept. Comput. Eng., Chalmers Univ. of Tech., Göteborg, Sweden, Sept. 2003.

[24] B. Andersson, S. Baruah, and J. Jonsson, "Static-priority scheduling on multiprocessors," in *Proc. IEEE Real-Time Systems Symp. (RTSS)*, London, UK, Dec. 2001, pp. 193–202.

[25] B. Andersson and J. Jonsson, "Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition," in *Proc. Int. Conf. Real-Time Computing Systems and Applications (RTCSA)*, Cheju Island, South Korea, Dec. 2000, pp. 337–346.

[26] T. P. Baker, "Multiprocessor EDF and deadline monotonic schedulability analysis," in *Proc. IEEE Real-Time Systems Symp. (RTSS)*, Cancun, Mexico, Dec. 2003, pp. 120–129.

[27] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing synchronous message deadlines with time token medium access control protocol," *IEEE Trans. on Comput.*, vol. 43, pp 327–339, Mar. 1994.

[28] N. Malcolm and W. Zhao, "Guaranteeing synchronous messages with arbitrary deadline constraints in an FDDI network," in *Proc. IEEE Conf. Local Computer Networks (LCN)*, Minneapolis MN, Sept. 1993, pp. 186–195.

[29] S. Zhang and A. Burns, "An optimal synchronous bandwidth allocation scheme for guaranteeing synchronous message deadlines with the timed-token MAC protocol," *IEEE/ACM Trans. Networking*, vol. 3, pp. 729–741, Dec. 1995.

[30] S. Zhang and E. S. Lee, "Efficient global allocation of synchronous bandwidths for hard real-time communication with the timed token MAC protocol," in *Proc. Int. Conf. Real-Time Computing Systems and Applications (RTCSA)*, Hong Kong, China, Dec. 1999, pp. 472–479.

[31] R. Sivakumar, T. Kim, N. Venkitaraman, and V. Bharghavan, "Achieving per-flow weighted rate fairness in a core stateless network," in *Proc. IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*, Taipei, Taiwan, April 2000, pp. 188–196.

[32] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit Differentiated Services architecture for the Internet," Network Working Group, RFC 2638, July 1999.

[33] R. Cruz, "SCED+: efficient management of quality of service guarantees," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1998, pp. 625–634.

[34] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *Proc. ACM SIGCOMM*, Cambridge, MA, Aug. 1999, pp. 81–94.

[35] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," Network Working Group, RFC 2212, Sep. 1997.

[36] A. Charny and J. Boudec, "Delay bounds in a network with aggregate scheduling," in *Proc. Int. Workshop Quality of Future Internet Services* (QOFIS), Berlin, Germany, Oct. 2000, pp. 1–13.

[37] C.-S. Chang, "Stability, queue length and delay of deterministic and stochastic queueing networks," *IEEE Trans. on Automatic Control*, vol. 39, pp. 913–931, May 1994.

[38] D. Starobinski and M. Sidi, "Stochastically bounded burstiness for communication networks," *IEEE Trans. Information Theory*, vol. 46, pp. 206–212, Jan. 2000.

[39] O. Yaron and M. Sidi, "Performance and stability of communication networks via robust exponential bounds," *IEEE/ACM Trans. Networking*, vol. 1, pp. 372–385, June 1993.

[40] J. Kurose, "On computing per-session performance bounds in high-speed multi-hop computer networks," in *Proc. ACM SIGMETRICS*, Newport, RI, June 1992, pp. 128–139.

[41] H. Zhang and E. Knightly, "Providing end-to-end statistical performance guarantees with bounding interval dependent stochastic models," in *Proc. ACM SIGMETRICS*, Nashville, TN, May 1994, pp. 211–220.

[42] J. Qiu and E. Knightly, "Inter-class resource sharing using statistical service envelopes," in *Proc. IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 1404–1411.

[43] S. Kweon and K. Shin, "Video-on-demand service using a statistical traffic envelope," Dept. Comput. Sci., Univ. of Michigan, Ann Arbor, Tech. Rep, 1998.

[44] E. W. Knightly, "Enforceable quality of service guarantees for bursty traffic streams," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1998, pp. 635–642.

[45] E. W. Knightly, "H-BIND: A new approach to providing statistical performance guarantees to VBR traffic," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 1091–1099.

[46] E. W. Knightly, "Second moment resource allocation in multi-service networks," in *Proc. ACM SIGMETRICS*, Seattle, WA, June 1997, pp. 181–191.

[47] D. Wu and R. Negi, "A wireless channel model for support of quality of service," in *Proc. IEEE GLOBECOM*, San Antonio, TX, Nov. 2001, pp. 695–699.

[48] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell System Technical Journal*, vol. 42, pp. 1977–1997, Sep. 1963.

[49] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, Sept 1960.

[50] B. Vucetic, "An adaptive coding scheme for time-varying channels," *IEEE Trans. on Communications*, vol. 39, pp. 653–663, May 1991.

[51] Q. Zhang and S. A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *IEEE Trans. on Communications*, vol. 47, pp. 1688–1692, Nov. 1999.

[52] H. S. Wang and N. Moayeri, "Finite-state Markov channel - a useful model for radio communication channels," *IEEE Trans. on Vehicular Technology*, vol. 45, pp. 258–264, May 1996.

[53] M. Krunz and J. G. Kim, "Fluid analysis of delay and packet discard performance for QoS support in wireless networks," *IEEE J. Select. Areas in Commun.*, vol. 19, pp. 384–395, Feb. 2001.

[54] Object Management Group. CORBA specifications. Accessed on Aug. 25, 2006. [Online]. Available: http://www.omg.org/technology/documents/corba_spec_catalog.htm.

[55] Microsoft. COM: delivering on the promises of component technology. Accessed on Aug. 25, 2006. [Online]. Available: http://www.microsoft.com/com.

[56] Sun Microsystems. Enterprise JavaBeans technology. Accessed on Aug. 25, 2006. [Online]. Available: http://java.sun.com/products/ejb.

[57] A. Pasetti and W. Pree, "The component software challenge for real-time systems," in *Proc. Int. Workshop on Real-Time Mission-Critical Systems*, Scottsdale, AZ, Nov.-Dec. 1999.

[58] Object Management Group, "Realtime CORBA joint revised submission," Document orbos/99-02-12 ed., Object Management Group, March 1999. Accessed on Aug. 25, 2006. [Online]. Available: http://www.omg.org/docs/orbos.

[59] D. C. Schmidt and F. Kuhns, "An overview of the real-time CORBA specification," *Computer*, vol. 33, pp. 56–63, June 2000.

[60] Real-time CORBA with TAO. Accessed on Aug. 25, 2006. [Online]. Available: http://www.cs.wustl.edu/~schmidt/TAO.html.

[61] B. Choi, S. Rho, and R. Bettati, "Fast software component migration for application survivability in distributed real-time systems," in *Proc. IEEE Int. Symp. on Object-oriented Real-time Distributed Computing (ISORC)*, Vienna, Austria, May 2004, pp. 269–276.

[62] J. A. Stankovic, R. Zhu, R. Poornalingam, C. Lu, Z. Yu, M. Humphrey, and B. Ellis, "VEST: An aspect-based composition tool for real-time systems," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)*, Toronto, Canada, May 2003, pp. 58–69.

[63] S. Wang, D. Xuan, R. Bettati, and W. Zhao, "Providing absolute differentiated services for real-time applications in static-priority scheduling networks," *IEEE/ACM Trans. Networking*, vol. 12, pp. 326-339, April 2004.

[64] R. Braden, D. Clark, S. Shenker, "Integrated services in the Internet architecture: an overview," Network Working Group, RFC 1633, June 1994. Accessed on Aug. 25, 2006. [Online]. Available: http://www.ietf.org/rfc/rfc1633.txt.

[65] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," Network Working Group, RFC 2638, July 1999. Accessed on Aug. 25, 2006. [Online]. Available: http://www.ietf.org/rfc/rfc2638.txt.

[66] R. Cruz, "A calculus for network delay, part I and part II," *IEEE Trans. Information Theory*, vol. 37, pp. 114–141, Jan. 1991.

[67] C. Li, R. Bettati, and W. Zhao, "Static priority scheduling for ATM networks," in *Proc. IEEE Real-Time Systems Symp. (RTSS)*, San Francisco, CA, Dec. 1997, pp. 264–273.

[68] C. Szyperski, D. Gruntz, and S. Murer, *Component Software: Beyond Object-Oriented Programming*, 2nd Edition. New York, NY: Addison-Wesley / ACM Press, 2002.

[69] A. Parekh, "A generalized processor sharing approach to flow control in integrated services networks," Ph.D. dissertation, Dept. of Electrical Eng. and Comput. Sci., MIT, Feb. 1992.

[70] C. Li, A. Raba, and W. Zhao, "Stability in ATM networks," in *Proc. IEEE INFOCOM*, Kobe, Japan, April 1997, pp. 160–167.

[71] D. Mitra and J. A. Morrison, "Erlang capacity and uniform approximations for shared unbuffered resources," *IEEE/ACM Trans. Networking*, vol. 2, pp. 558–570, Dec. 1994.

[72] S. Wang, R. Nathuji, R. Bettati and W. Zhao, "Real-time guarantees in wireless networks," in *Resource Management in Wireless Networking*, M. Cardei, I. Cardei and D.-Z. Du (Eds.), Dordrecht, The Netherlands, Springer, 2005.

[73] J. G. Proakis, *Digital Communications.* New York, NY: McGraw-Hill, 1995.

[74] J. B. Cain and D. N. McGregor, "A recommended error control architecture for ATM networks with wireless links," *IEEE J. Select. Areas in Commu.*, vol. 15, pp. 16–28, Jan. 1997.

[75] R. Fantacci, "Queuing analysis of the selective repeat automatic repeat request protocol wireless packet networks," *IEEE Trans. Veh. Technol.*, vol. 45, pp. 258–264, May 1996.

[76] P. Bello, "Aeronautical channel characterization," *IEEE Trans. Commun.*, vol. 21, pp. 548–563, May 1973.

[77] J. Hagenauer and W. Papke, "Data transmission for maritime and land mobile using stored channel simulation," in *Proc. IEEE Veh. Technol. Conf.*, San Diego, CA, USA, pp. 379–383, May 1982.

[78] R. W. Huck, J. S. Butterworth, and E. E. Matt, "Propagation measurements for land mobile satellite services," in *Proc. Veh. Technol. Conf.*, Toronto, Canada, pp. 265–268, May 1983.

[79] M. Schwartz, *Broadband Integrated Networks*. Upper Saddle River, NJ: Prentice Hall, 1996.

[80] S. Lin and Jr. D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Upper Saddle River, NJ: Prentice Hall, 1983.

[81] Modeling topology of large internetworks. Accessed on Aug. 25, 2006. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm

[82] D. Wrege, E. Knightly, H. Zhang, and J. Liebeherr, "Deterministic delay bounds for VBR video in packet switching networks: fundamental limits and practical tradeoffs," *IEEE/ACM Trans. Networking*, vol. 4, pp. 352-362, June 1996.

[83] F. P. Kelly, "Blocking probabilities in large circuit-switched networks," *Advance in Applied Probability*, vol. 18, pp. 473–505, June 1986.

[84] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan, "ATM network design and optimization: a multirate loss network framework," *IEEE/ACM Trans. Networking*, vol. 4, pp. 531–543, 1996.

[85] JBoss. Accessed on Aug. 25, 2006. [Online]. Available: http://www.jboss.org.

[86] Sun Microsystems. Java Management Extensions. Accessed on Aug. 25, 2006. [Online]. Available: http://java.sun.com/products/JavaManagement.

[87] M. Fleury and F. Reverbel, "The JBoss extensible server," in *Proc. ACM/IFIP/USENIX Int' Middleware Conf. (Middleware)*, Rio de Janeiro, Brazil, June 2003, pp. 344–373.

[88] S. Stark and The JBoss Group. JBoss Administration and Development Documentation – ebook – 3.2.1. Accessed on Aug. 25, 2006. [Online]. Available: http://www.jboss.org/docs/index.

[89] Sun Microsystems. Java Remote Method Invocation Specification. Accessed on Aug. 25, 2006. [Online]. Available: http://java.sun.com/j2se/1.3/docs/guide/rmi.

[90] TimeSys. TimeSys Linux 3.1. Accessed on Aug. 25, 2006. [Online]. Available: http://www.timesys.com.

[91] TimeSys. Real-time specification for Java reference implementation, 2001, Accessed on Aug. 25, 2006. [Online]. Available: http://www.timesys.com.

# APPENDIX A

## PROOF OF THEOREM III-2

**Theorem III-2** includes two main results: one is the aggregation function, and the other is the worst-case delay bound. To prove these results, in the following, we introduce two lemmas: **Lemma A-1** tries to get the expression of the aggregation function and **Lemma A-2** tries to derive the worst-case delay.

**Lemma A-1.** The aggregated traffic of group $G_{p,j,k}$ is constrained by (III-23).

*Proof.* For any flow $x$ in group $G^i_{p,j,k}$, let $Y_x$ be the total worst-case queueing delay experienced by any packets in flow $x$ upstream from Server $k$. Suppose that $Y^i_{p,k}$ is the maximum of the worst-case queueing delays $Y_x$:

$$Y_x \le Y^i_{p,k}. \tag{A-1}$$

Since $H^i(I)$ is the source traffic function of flow $x$, according to Theorem 2.1 in [66], we have

$$F_x(I) \le H^i(I + Y_x) \le H^i(I + Y^i_{p,k}). \tag{A-2}$$

Recall that the source traffic function is $H^i(I) \le \sigma^i + \rho^i I$. Therefore, by (A-2), we can bound $F^i_{p,j,k}$ as follows:

$$\begin{aligned} F^i_{p,j,k}(I) &\le \sum\nolimits_{x \in G^i_{p,j,k}} F_x(I) \\ &\le \sum\nolimits_{x \in G^i_{p,j,k}} H^i(I + Y^i_{p,k}) \\ &\le n^i_{p,j,k}(\eta^i_{p,k} + \rho^i I). \end{aligned} \tag{A-3}$$

On the other hand, the total amount of traffic that can be transmitted over input link $j$ of Server $k$ during any time interval $I$ is constrained by the link capacity $C$, i.e.,

$$F^i_{p,j,k}(I) \le C_{j,k} I. \tag{A-4}$$

Synthesizing (A-3) and (A-4), we have

$$F^i_{p,j,k}(I) = \begin{cases} C_{j,k} I, \mathrm{I} \le \tau^i_{p,j,k} \\ \mathrm{n}^i_{p,j,k}(\eta^i_{p,k} + \rho^i I), \mathrm{I} > \tau^i_{p,j,k} \end{cases} \tag{A-5}$$

where

$$\tau^i_{p,j,k} = \frac{n^i_{p,j,k}\eta^i_{p,k}}{C_{j,k} - n^i_{p,j,k}\rho^i}. \tag{A-6}$$

Furthermore, bounds can be defined for the aggregated traffic of group $G_{p,j,k}$ as follows:

$$F_{p,j,k}(I) = \min\{C_{j,k} I, \textstyle\sum_{i=1}^{M} F^i_{p,j,k}(I)\}. \tag{A-7}$$

Notice that $\tau_{p,j,k} \ge \tau^i_{p,j,k}$ for all classes $i$. $\sum_{i=1}^{M} F^i_{p,j,k}(I) \ge C_{j,k} I$ as $I \le \tau_{p,j,k}$ and $\sum_{i=1}^{M} F^i_{p,j,k}(I) = n_{p,j,k} \cdot (\eta_{p,k} + \rho I)$ as $I > \tau_{p,j,k}$. Thus, (III-23) holds as claimed. □

Recall that given a static-priority scheduling discipline at the server, we have the following formula that indicates how long a newly-arrived priority-*p* packet can be delayed at Server *k* [67]:

$$d_{p,k} = \frac{1}{C_k} \max_{I < I_{p,k}} (F_{p,k}(I + d_{p,k}) - C_k I), \tag{A-8}$$

where

$$F_{p,k}(I+d_{p,k})=\sum_{q=1}^{p-1}\sum_{j=1}^{L_k}F_{q,j,k}(I+d_{p,k})+\sum_{j=1}^{L_k}F_{p,j,k}(I), \tag{A-9}$$

$$I_{p,k}=\min\{I>0:\sum_{q=1}^{p}\sum_{j=1}^{L_k}F_{q,j,k}(I)\le C_kI\}. \tag{A-10}$$

Applying (III-23) into (A-9), we can obtain the explicit expression of $F_{p,k}(I+d_{p,k})$ in (A-9). To get the explicit expression of $d_{p,k}$, we need to find the point where the worst-case delay in (A-8) is suffered, and then we can remove the maximum symbol. The following lemma tries to address this issue:

**Lemma A-2.** Define $\tilde{\tau}$ as the point after which the slope of the aggregate traffic function becomes smaller than $C_k$, and then the worst-case queuing delay $d_{p,k}$ suffered by the traffic with priority $p$ at Server $k$ will happen at

$$I=\tilde{\tau}=\frac{n_{p,j',k}\cdot\eta_{p,k}}{C_{j',k}-n_{p,j',k}\cdot\rho}, \tag{A-11}$$

for some specific $j'$, where $1\le j'\le L_k$.

*Proof.* Note that each $F^i_{p,j,k}(I)$ is a two-piece-wise linear continuous function, and $\sum_{i=1}^{M}F^i_{p,j,k}(I)$ is still a piece-wise linear continuous function. The value $\tau^i_{p,j,k}$ identifies the intersection of the two linear segments, and is called the *flex point* of $F^i_{p,j,k}(I)$. All $\tau^i_{p,j,k}$'s are also flex points of $\sum_{i=1}^{M}F^i_{p,j,k}(I)$. We know that all traffic constraint functions $F_{q,j,k}(I)$'s are piece-wise, and so is the aggregated traffic constraint functions $F_{p,k}(I+d_{p,k})$. We try to find the flex point $\tilde{\tau}$ so that this flex point maximizes the delay in (A-8) (as shown in Figure A-1).

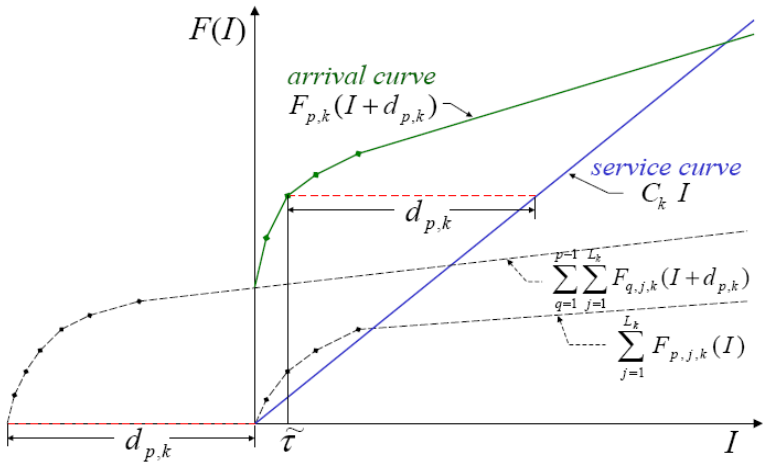



Figure A-1. The Detailed Illustration of Arrival Curve $F_{p,k}(I+d_{p,k})$

The aggregated traffic function consists of two parts: the aggregated traffic function with priorities higher than $p$ and the one with priority $p$.

- *The aggregated traffic function with priorities higher than $p$:* Define $\tau_{p-,k}$ as the maximum of the flex points of the traffic constraint function $F_{q,j,k}(I)$ with priority $q < p$. Let $I_{p-,k}$ be the maximum busy interval of the aggregated traffic constraint function $\sum_{q=1}^{p-1}\sum_{j=1}^{L_k} F_{q,j,k}(I)$. As we know $d_{p,k} \geq I_{p-,k} \geq \tau_{p-,k}$. Therefore, the aggregated traffic function $\sum_{q=1}^{p-1}\sum_{j=1}^{L_k} F_{q,j,k}(I+d_{p,k})$ has no flex points as $I \geq 0$.
- *The aggregated traffic function with priority $p$:* Let $\tau_{p,j,k}$ be the flex point of the traffic constraint function $F_{p,j,k}(I)$. It is also the flex point of $F_{p,k}(I+d_{p,k})$. As we know $I_{p,k} \geq \tau_{p,j,k}$ for all $j$'s.

Therefore, the worst-case queuing delay $d_{p,k}$ suffered by the traffic with priority $p$ at Server $k$ will happen at

$$I = \tilde{\tau} = \frac{n_{p,j',k} \cdot \eta_{p,k}}{C_{j',k} - n_{p,j',k} \cdot \rho}, \tag{A-12}$$

for some specific $j'$, where $1 \le j' \le L_k$. □

Now, we are ready to prove **Theorem III-2**.

*Proof.* (III-23) can be obtained directly from **Lemma A-1** Let us derive (III-26). By **Lemma A-2**, we know that the worst-case queuing delay $d_{p,k}$ suffered by the traffic with priority $p$ at Server $k$ will happen at $I = \tilde{\tau} = \frac{n_{p,j',k} \cdot \eta_{p,k}}{C_{j',k} - n_{p,j',k} \cdot \rho}$ for some specific $j'$, where $1 \le j' \le L_k$. Substituting (A-5) and (A-11) into (A-8), we can, therefore, eliminate the max operator from (A-8) as follows:

$$d_{p,k} \le \frac{1}{C_k} \left( \sum_{q=1}^{p-1} \sum_{j=1}^{L_k} n_{q,j,k} \cdot (\eta_{q,k} + \rho(\tilde{\tau} + d_{p,k})) + \sum_{j=1}^{L_k} n_{p,j,k} \cdot (\eta_{p,k} + \rho\tilde{\tau}) \right). \tag{A-13}$$

By appropriately redefining some parameters and some algebraic manipulation in (A-13), we have

$$d_{p,k} \le \frac{U_{p,k} - V_{p,k} W_{p,k}}{X_{p,k}}, \tag{A-14}$$

where $U_{p,k}$, $V_{p,k}$, $W_{p,k}$, and $X_{p,k}$ are defined in (III-27) - (III-30), respectively. □

# APPENDIX B

## PROOF OF THEOREM III-3

In order to prove **Theorem III-3**, we need two lemmas.

Before presenting these lemmas, we define the worst-case of arrival curve $\tilde{F}_{p,k}(I+d_{p,k})$ as the case that, in the arrival curve $F_{p,k}(I+d_{p,k})$, $\sum_{j=1}^{L_k} F_{p,j,k}(I)$ becomes

$$\tilde{F}_{p,k}(I)=\min\{\sum_{j=1}^{L_k} C_{j,k}I, \sum_{j=1}^{L_k} n_{p,j,k}\cdot(\eta_{p,k}+\rho I)\}, \tag{B-1}$$

then let us introduce the first lemma:

**Lemma B-1.** The worst-case queuing delay at Server $k$ suffered by any packet with priority $p$ can be experienced if the arrival curve becomes the worst-case $\tilde{F}_{p,k}(I+d_{p,k})$, and furthermore we have

$$W_{p,k}=\tilde{\tau}=\frac{\sum_{j=1}^{L_k} n_{p,j,k}\cdot\eta_{p,k}}{\sum_{j=1}^{L_k}(C_{j,k}-n_{p,j,k}\cdot\rho)}, \tag{B-2}$$

where $W_{p,k}$ is defined in (III-30).

*Proof.* Note that given all specific $n_{q,j,k}$'s, $\sum_{j=1}^{L_k} F_{p,j,k}(I)\le\tilde{F}_{p,k}(I)$ for general $F_{p,j,k}(I)$, therefore, $F_{p,k}(I+d_{p,k})\le\tilde{F}_{p,k}(I+d_{p,k})$ for general $F_{p,k}(I+d_{p,k})$ as illustrated in Figure B-1. By (A-8), we know that the larger $F_{p,k}(I+d_{p,k})$, the larger $d_{p,k}$. Therefore, the worst-case arrival curve will experience the worst-case delay. Note that $\tilde{\tau}$ defined in (B-2) is the only flex point of the function (B-1). Therefore, the worst-case delay will happen at $I=\tilde{\tau}$, and then $W_{p,k}=\tilde{\tau}$.

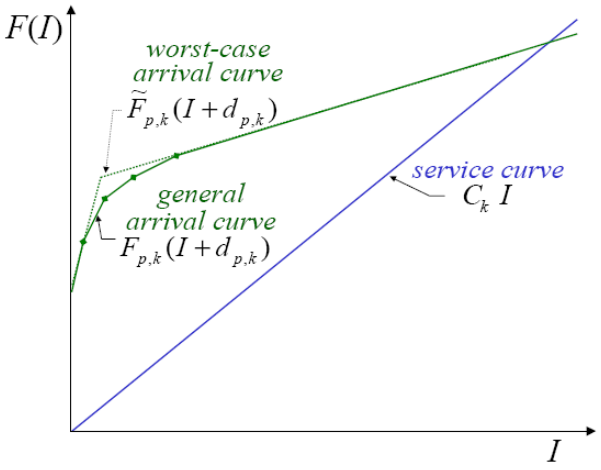



Figure B-1. Worst-case Arrival Curve $\tilde{F}_{p,k}(I+d_{p,k})$

On the other hand, we show that the worst-case arrival curve can be achieved for some specific $n_{p,j,k}$'s. Note that if all flex points $\tau_{p,j,k}(I)$'s of $F_{p,j,k}(I)$'s are equal, *i.e.*,

$$\tau_{p,1,k} = \tau_{p,2,k} = \ldots = \tau_{p,L_k,k}. \tag{B-3}$$

Applying the formula

$$\frac{x_1}{y_1} = \frac{x_2}{y_2} = \cdots = \frac{x_{L_k}}{y_{L_k}} = \frac{x_1 + x_2 + \cdots + x_{L_k}}{y_1 + y_2 + \cdots + y_{L_k}} \tag{B-4}$$

to (B-3), we have

$$\tau_{p,1,k} = \tau_{p,2,k} = \ldots = \tau_{p,L_k,k} = \tilde{\tau}. \tag{B-5}$$

That means $\sum_{j=1}^{L_k} F_{p,j,k}(I)$ becomes $\tilde{F}_{p,k}(I)$, and, hence, the arrival curve $F_{p,k}(I+d_{p,k})$ becomes the worst-case arrival curve $\tilde{F}_{p,k}(I+d_{p,k})$. For this worse-case,

$$W_{p,k} = \tilde{\tau} = \frac{\sum_{j=1}^{L_k} n_{p,j,k} \cdot \eta_{p,k}}{\sum_{j=1}^{L_k} (C_{j,k} - n_{p,j,k} \cdot \rho)}. \quad \text{(B-6)}$$

□

**Lemma B-2.** The worst-case queuing delay at Server $k$ suffered by any class-$i$ packet with priority $q$ is experienced only if the number of flows $n_{q,k}^i$ is maximized, *i.e.*,

$$n_{q,k}^i = \gamma_{q,k}^i C_k, \quad \text{(B-7)}$$

where

$$\gamma_{q,k}^i = \frac{\alpha_{q,k}^i}{\rho^i}. \quad \text{(B-8)}$$

*Proof.* By (A-8), we know that the larger $F_{q,j,k}(I)$, the larger $d_{p,k}$. Furthermore, since $F_{q,j,k}(I)$ is the aggregated class-$i$ traffic with priority $q$ at Server $k$, we know that the larger $n_{q,k}^i$, the larger $F_{q,j,k}(I)$. Therefore, when the number of flows on each link is maximized, then any class-$i$ packet with priority $p$ will experience the worst-case queuing delay at the server, *i.e.*,

$$n_{q,k}^i = \gamma_{q,k}^i C_k. \quad \text{(B-9)}$$

In general, $\gamma_{q,k}^i C_k$ is not necessarily an integer. However, in a modern practical system, it is very large, and we can assume that $\lfloor \gamma_{q,k}^i C_k \rfloor \approx \gamma_{q,k}^i C_k$. For example, if we consider a Gigabit router, $C_k = 1 \times 10^9$ bps, for voice traffic $\rho^i = 32,000$ bps, if $\alpha_{q,k}^i = 15\%$, then $\gamma_{p,k}^i C = 4,687.5$. □

Now, we are ready to prove **Theorem III-3**.

*Proof.* By **Lemma B-1** and **Lemma B-2**, we obtain (B-7) and (B-2). Substituting (B-7) into (III-27)--(III-29) and (B-2), we have

$$U_{p,k} \le (\sum_{q=1}^{p} \gamma_{q,k} \cdot \eta_{q,k}) C_k \tag{B-10}$$

$$V_{p,k} \ge C_k - (\sum_{q=1}^{p} \gamma_{q,k} \cdot \rho) C_k \tag{B-11}$$

$$X_{p,k} \ge C_k - (\sum_{q=1}^{p-1} \gamma_{q,k} \cdot \rho) C_k \tag{B-12}$$

and

$$W_{p,k} \ge \frac{(\gamma_{p,k} \cdot \eta_{p,k}) C_k}{\sum_{j=1}^{L_k} C_{j,k} - (\gamma_{p,k} \cdot \rho) C_k}. \tag{B-13}$$

since $\gamma_{q,k} \cdot \eta_{q,k} = \alpha_{q,k} \cdot Z_{q,k}$, $\gamma_{q,k} \cdot \rho = \left\| \alpha_{q,k} \right\|$, and $c_k = \frac{1}{C_k} \sum_{j=1}^{L_k} C_{j,k}$, $U_{p,k}, V_{p,k}, X_{p,k}$ and $W_{p,k}$ can be verified as claimed. □

# APPENDIX C

## PROOF OF COROLLARY III-1

*Proof.* Recall that $d$ is the maximum of worst-case delays suffered by all real-time class packets across all link servers in the network, and $Y_k$ is the maximum of worst-case delays suffered by any real-time class packet upstream from Server $k$, therefore,

$$Y_k \le (h-1)d. \tag{C-1}$$

On the other hand, by (III-3), we have

$$d \le r(\frac{\sigma}{\rho} + Y_k). \tag{C-2}$$

Combining (C-1) and (C-2), as $\frac{1}{r} > (h-1)$, i.e., $\alpha < \frac{1}{1+(h-2)(1-\frac{1}{L})}$, we have

$$d \le \frac{1}{\frac{1}{r}-(h-1)}\frac{\sigma}{\rho}. \tag{C-3}$$

Furthermore, the maximum end-to-end delay can be bounded as follows:

$$d^{e2e} \le h\,d \le \frac{h}{\frac{1}{r}-(h-1)}\frac{\sigma}{\rho}. \tag{C-4}$$

□

# APPENDIX D

## PROOF OF COROLLARY III-2

*Proof.* (Little abusing parameters $\hat{d}_k$ and $Y_k$) We also define $\hat{d}_k$ as the worst-case delays bound suffered by any real-time class packet at layer-$k$ link servers, and $Y_k$ as the worst case queuing delay bound suffered by any real-time class packet upstream from layer-$k$ link server, i.e.,

$$Y_k = \sum_{l=1}^{k-1} \hat{d}_l. \tag{D-1}$$

By (III-3), we have

$$\hat{d}_k = r(\frac{\sigma}{\rho} + Y_k). \tag{D-2}$$

Therefore, by (D-1) and (D-2), we have

$$\hat{d}_k - \hat{d}_{k-1} = r\hat{d}_{k-1}, \tag{D-3}$$

and then,

$$\hat{d}_k = (r+1)\hat{d}_{k-1} = \cdots = (r+1)^{k-1}\hat{d}_1. \tag{D-4}$$

We know $\hat{d}_1 = r\frac{\sigma}{\rho}$, therefore, $\hat{d}_k$, the maximum of worst-case delays suffered by any real-time class packet at layer-$k$ link servers, can be bounded as follow:

$$\hat{d}_k \le r(r+1)^{k-1}\frac{\sigma}{\rho}, \tag{D-5}$$

and $Y_k$, the maximum of worst-case delays suffered by any real-time class packet upstream from layer-$k$ link server, can be bounded as follow:

$$Y_k \le \sum_{l=1}^{k-1} \hat{d}_l \le \sum_{l=1}^{k-1} r(r+1)^{l-1} \frac{\sigma}{\rho} = ((r+1)^{k-1} - 1) \frac{\sigma}{\rho}. \quad \text{(D-6)}$$

Therefore, the maximum end-to-end delay can be bounded as follows:

$$d^{e2e} \le Y_{\hat{h}+1} \le ((r+1)^{\hat{h}} - 1) \frac{\sigma}{\rho}. \quad \text{(D-7)}$$

□

# APPENDIX E

## PROOF OF THEOREM IV-2

The following lemmas [44] define the mean rate for a group of flows and show how an upper bound on the stochastic rate variance envelope can be derived from the deterministic parameters.

**Lemma E-1.** (Mean Rate) The mean rate of the group of flows $G_{i,j}$ can be defined as:

$$\phi_{i,j} = \lim_{I \to \infty} \frac{F_{i,j}(I)}{I}. \tag{E-1}$$

**Lemma E-2.** (Adversarial Mode) The rate-variance envelope of the group of flows $G_{i,j}$ is upper bounded by:

$$RV_{i,j}(I) \le RV^{*}_{i,j}(I) = \phi_{i,j}\left(\frac{F_{i,j}(I)}{I} - \phi_{i,j}\right) \tag{E-2}$$

where $\phi_{i,j}$ is defined in (E-1).

**Lemma E-3.** (Non-adversarial Mode) The rate-variance envelope of the group of flows $G_{i,j}$ is approximately:

$$RV_{i,j}(I) \approx \widehat{RV}_{i,j}(I) = \frac{\phi_{i,j}}{12}\left(\frac{F_{i,j}(I)}{I} - \phi_{i,j}\right) \tag{E-3}$$

where $\phi_{i,j}$ is defined in (E-1).

We know that the aggregated arrival traffic constraint function $F_{i,j}(I)$ is given as follows

$$F_{i,j}(I) = \begin{cases} C \cdot I, I \le \tau_{i,j} \\ n_{i,j}(\sigma_i + \rho_i\ I), I > \tau_{i,j} \end{cases} \quad \text{(E-4)}$$

Then applying (E-4) to (E-1), (E-2), and (E-3), **Theorem IV-2** can be proved as claimed.

# APPENDIX F

# SERVICE OPTIMIZATION

In (VI-9), the rejection probability $b_r$ is not determined yet, and we compute it in the following using Kelly's approximation approach [83].

We first compute the resource need of different Tasks. We do this with help of a reference unit resource. Let the parameters of a unit resource be $\langle \sigma, \rho \rangle$.[10] Define $\langle \sigma_{r,e,i}, \rho_{r,e,i} \rangle$ as the arrival constraint function of any task along Task Route $r$ at Component $e$ of Class $i$, then

$$\sigma_{r,e,i} = \sigma_r + \rho_r \sum_{\langle e_h, i_h \rangle \prec_r \langle e,i \rangle} D_{e_h,i_h}, \tag{F-1}$$

$$\rho_{r,e,i} = \rho_r, \tag{F-2}$$

where $\langle e_h, i_h \rangle \prec_r \langle e, i \rangle$ denotes all component services $\langle e_h, i_h \rangle$'s along Task Route $r$ before component service $\langle e, i \rangle$. The resource for $\langle \sigma_{r,e,i}, \rho_{r,e,i} \rangle$ can therefore be represented by the scale $a_{r,e,i} = \min\{\lfloor \sigma_{r,e,i}/\sigma \rfloor, \lfloor \rho_{r,e,i}/\rho \rfloor\}$. Similarly, the resource at Component $e$ of Class $i$ is represented by the scale $a_{e,i} = \min\{\lfloor \sigma_{e,i}/\sigma \rfloor, \lfloor \rho_{e,i}/\rho \rfloor\}$.

Define rejection probability $b_{e,i}$ as the probability that a task request for Component $e$ of Class $i$ is rejected. Under the assumption that the rejections at all components are independent [83], we have

---

[10] $\langle \sigma, \rho \rangle$ can be chosen as the greatest common divisor (g.c.d) of $\sigma_r$'s and $\rho_r$'s, respectively.

$$b_r = 1 - \prod_{\langle e,i \rangle \in r} (1 - b_{e,i})^{a_{r,e,i}}. \tag{F-3}$$

By *Erlang's Formula*, $b_{e,i}$ is given by

$$b_{e,i} = E[a_{e,i}; \nu_{e,i}], \tag{F-4}$$

where $E[a;\nu] = (\nu^a/a!)/(\sum_{n=0}^{a} \nu^n/n!)$, and $\nu_{e,i}$ is the admitted load to Component $e$ of Class $i$. With the independence assumption, $\nu_{e,i}$ is given by

$$\nu_{e,i} = \frac{1}{1 - b_{e,i}} \sum_{r \in R_{e,i}} a_{r,e,i} \nu_r (1 - b_r), \tag{F-5}$$

where $R_{e,i}$ is the set of task routes that go through Component $e$ of Class $i$.

Therefore, $b_r$ is a solution to the fixed point equations (F-3), (F-4) and (F-5). Hence, $AP$ in (VI-9) can be obtained. This leads to the solution of the optimization problem. The objective function of the optimization problem is nonlinear while all the constraints are linear. Therefore, the optimization is a *linearly-constrained optimization*. There are two issues involving in solving the optimization problem:

- Minimum operators appear in both $a_{e,i}$ and $a_{r,e,i}$. From (F-1) and (F-2), we find that the burst size will increase along the task route, but the average rate will not. Therefore, during optimization process, we can set $a_{e,i} = \lfloor \rho_{e,i}/\rho \rfloor \le \lfloor \sigma_{e,i}/\sigma \rfloor$ and $a_{r,e,i} = \lfloor \rho_{r,e,i}/\rho \rfloor \le \lfloor \sigma_{r,e,i}/\sigma \rfloor$. Minimum operators can be removed.
- The objective function is not continuous because $a_{e,i}$ and $a_{r,e,i}$ are integer functions, which make the problem even harder. If $a_{e,i}$ is small, we can use

exhaustive search of $a_{e,i}$ to find the optimal value. Otherwise, we can reset $a_{e,i} = \rho_{e,i}/\rho \le \sigma_{e,i}/\sigma$ and $a_{r,e,i} = \rho_{r,e,i}/\rho \le \sigma_{r,e,i}/\sigma$, and approximate $b_r$ with the uniform asymptotic approximation (UAA) method [84]. Then the objective function becomes continuous and it can be solved by an optimization toolbox.

In the above, we assume that the application arrival pattern can be predicted *a priori*. Otherwise, it may be hard or impossible to specify good real-time services of components beforehand. However, the application pattern can be monitored provided that the application pattern in our system will not change frequently. Based on the observed application pattern, an optimal service specification can gradually be achieved. Periodically updating the service specifications to reflect changes in the application patterns can greatly increase the utilization level of resources.

# VITA

Shengquan Wang received a B.S. degree in mathematics from Anhui Normal University, China, in 1995, an M.S. degree in applied mathematics at Shanghai Jiao Tong University, China, in 1998, and an M.S. degree in mathematics at Texas A&M University in 2000. He began his doctoral studies in computer science at Texas A&M University in 2000 and received a Ph.D. degree in 2006. His research interests are in the areas of networking and distributed systems with a primary focus on real-time computing and communication, security, privacy, and reliability. He can be reached care of Dr. Wei Zhao, Professor, Department of Computer Science, Texas A&M University, College Station, TX, 77843-3112.