ABSTRACT

ANJUM, BUSHRA. Bandwidth Allocation under End-to-End Percentile Delay Bounds. (Under the direction of Harry Perros).

The main objective of this thesis is the estimation of the bandwidth that should be allocated on each link along the path of a flow of IP packets, so that the end-to-end delay D is bounded statistically. That is, D is less than or equal to a given value T with a probability γ . The path through an IP network is depicted by a tandem queueing network of infinite capacity queues, where each queue represents the output port of a router.

We first obtain some results on adding percentiles, and then we propose two different algorithms for calculating end-to-end percentiles, which is then used in a simple search algorithm to obtain the required bandwidth. The first algorithm is based on probabilistic assumptions regarding the arrival process of the packet flow and service times at each node in the tandem queueing network. The second algorithm is based on traces, such as video traces, that consist of a sequence of tuples, each describing the time of arrival of a packet and its packet length.

In the first algorithm, we assume that the arrival process to the first queue of the queueing network is bursty and correlated and the service times are exponentially distributed. Initially, we assumed an MMPP (Markov-modulated Poisson process) arrival process, and then we extended the algorithm to a MAP2 (two-stage Markov arrival process) arrival process. A MAP can represent a variety of processes that includes, as special cases, the Poisson process, the phase-type renewal processes, the MMPP and superposition of these. No background arrivals were considered at each node, the implication being that the output port scheduler is assumed to be per flow. The proposed algorithm is based on an interpolation between an upper and lower bound obtained by analyzing only the first queue. It is a relatively simple algorithm despite the complexity of the problem, and as shown through extensive comparisons with simulation data that it has a very low relative error. The average error is 1.25% for MMPP and 4.5% for MAP2.

Video traffic is widely expected to account for a large portion of the traffic in future wired and wireless networks. For interactive video the end-to-end delay has to be less than

150 to 200 msec in order to guarantee good QoE. Interestingly enough, little is known about how much bandwidth should be allocated to an interactive video so that a given percentile of the end-to-end delay is satisfied. Using the above algorithm, we obtained the end-to-end percentile for video traffic, by approximating a video trace with a MAP2. Subsequently, the required bandwidth so that a given end-to-end percentile is satisfied is obtained using a simple search algorithm. Three different types of video traces were used, namely, Cisco's Telepresence, IPTV and WebEx. The approximation of video trace by a MAP2 coupled with the above algorithm gives results with a low relative error, the average relative error is around 6%.

The algorithm described above for calculating a percentile of the end-to-end delay of a video stream, cannot handle the presence of background traffic that competes with the video within the output port of a router. In view of this, we propose an exact and efficient algorithm for calculating any percentile of the end-to-end delay in the presence of background video traffic, assuming that all video flows are characterized by packet traces and an arriving packet keeps its packet length in the entire queueing network. These traces may be a single stream or multiple streams. Packets in each node are served in a FIFO, the implication being that the output port scheduler is assumed to be per class, which more general than the assumption made in the first algorithm. A class could be a DiffServ class where all video packets are queued. Based on this algorithm, the minimum amount of bandwidth required, so that a given percentile of the end-to-end delay is satisfied, is easily obtained using a simple search algorithm.

Bandwidth Allocation under End-to-End Percentile Delay Bounds

by
Bushra Anjum

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2012

APPROVED BY:

Dr. William Stewart

Dr. David Thuent

Dr. Harry Perros
Chair of Advisory Committee

Dr. Michael Devetsikiotis

DEDICATION

To Mom!

Hope you are looking down from the heaven skies and smiling proudly at me!

BIOGRAPHY

Bushra Anjum is a Fulbright scholar and is currently a PhD student at North Carolina State University, advised by Dr. Harry Perros. She received her Master's of Science in 2007, maintaining rank 1 in a batch of 250 students and earning the Gold Medal, from the Computer Science Department at the Lahore University of Management Sciences, Pakistan. She received her Bachelor's of Science, Summa Cum Laude, from the Computer Science Department at National University of Computer and Emerging Sciences, Pakistan. She has authored scholarly papers in the areas of performance modeling, computer networks, and quality of service provision.

ACKNOWLEDGEMENTS

Dr. Harry Perros & Dr. Helen Perros for unfaltering academic, moral and emotional support, committee members (Dr. David Thunte, Dr. Michael Devetsikiotis and Dr. William Stewart) for their guidance and advice, my father and sisters for their unending faith in me, Fulbright & NC State for granting me this wonderful opportunity to study abroad, the Pierces (especially Kris Pierce) for being my family away from family, Kristi Brooks and her wonderful family and precious friends like Pam Cook, Fiva McCanless, Mahwash Pervaiz, Khurram Pervaiz, Maliha Nazir, Ayesha Afzal, Saima Mushtaq, Sana Abid, Pavithra Ramaswamay, the Ackland family and the Klar family, my support circle!

TABLE OF CONTENTS

LIST OF TABLES.....	ix
LIST OF FIGURES.....	x
Chapter 1	1
1 Introduction	1
1.1 End-to-End Quality of Service.....	1
1.2 IP Network QoS Architectures.....	2
1.2.1 Real Time Protocol (RTP).....	2
1.2.2 Integrated Services (IntServ).....	3
1.2.3 Differentiated Services (Diffserv).....	4
1.2.4 Traffic Engineering	6
1.2.5 Multiprotocol label switching (MPLS)	7
1.2.6 MPLS Overlay Networks	8
1.2.7 Queue Management and Scheduling Policies	9
1.3 End-to-end Bandwidth Guarantees	10
1.3.1 Bandwidth allocation based on Packet Loss	10
1.3.2 Bandwidth allocation based on end-to-end delay	12
1.4 Summary of the Thesis.....	16
Chapter 2	19
2 Percentile Arithmetic	19
2.1 Introduction	19
2.2 Problem Statement	21
2.3 Calculation of the Weight Function	22

2.3.1	Exponential Components with identical rate parameter	22
2.3.2	Exponential Components with different rate parameter	29
2.3.3	Two-stage Coxian	35
2.4	Inter-provider Quality of Service	40
2.5	Calculating Percentiles from Actual Latency Data	45
2.6	Single Source Shortest Path using Dijkstra's Algorithm	48
2.7	Conclusions	50
Chapter 3		52
3	Bandwidth Allocation for an MMPP2 Arrival Process	52
3.1	Introduction	52
3.2	The queueing network under study	55
3.3	The queueing network under study	57
3.4	Bandwidth estimation based on bounds	59
3.5	Validation	64
3.6	Conclusions	70
Chapter 4		72
4	Bandwidth Allocation for a MAP2 Arrival Process	72
4.1	Introduction	72
4.2	The queueing network under study	76
4.3	End-to-end delay estimation based on bounds	78
4.3.1	The Interpolation Function.....	80
4.4	Validation	82
4.5	Video Traces	85
4.6	Conclusions	90

Chapter 5 92

5	Bandwidth Allocation of a Video Stream using Traces	92
5.1	Introduction	92
5.2	Literature review	93
5.3	The Tandem Queueing Network under Study.....	95
5.4	The Proposed Algorithm: No Background Arrivals	96
5.5	Bandwidth Requirement for Homogeneous Flows	99
5.5.1	Telepresence.....	100
5.5.2	WebEx.....	104
5.5.3	IPTV	108
5.5.4	Linear Bandwidth Gain	111
5.6	Analysis of the Tandem Queueing Network with Background Traffic	117
5.7	Jitter and Delay Percentile Analysis.....	118
5.7.1	Jitter	119
5.7.2	Results	119
5.7.3	Telepresence.....	120
5.7.4	WebEx.....	121
5.8	Bandwidth Allocation under Jitter and Percentile Delay Bounds.....	123
5.8.1	Telepresence.....	124
5.8.2	WebEx.....	127
5.8.3	IPTV	130
5.9	Conclusions	132

Chapter 6 134

6 Conclusion and Future Work..... 134

LIST OF TABLES

Table 1.1 Scheduling Policies	9
Table 2.1 Population statistics in ms.....	46
Table 2.2 A comparison of percentiles	48
Table 3.1 Squared-coefficient of variation of the departure process.....	65
Table 3.2 Relative Errors	69
Table 5.1 Statistics of Multiplexed Telepresence Traces	101
Table 5.2 Statistics of Resultant Trace as multiple WebEx traces get multiplexed together	105
Table 5.3 Statistics of Resultant Trace as multiple IPTV traces get multiplexed together ..	108
Table 5.4 Simulation vs Algorithm CPU Time	118

LIST OF FIGURES

Figure 2.1 Composition of n components	22
Figure 2.2 A sequence of exponential distribution	23
Figure 2.3 Erlang- n versus sum of n exponentials.....	24
Figure 2.4 Erlang- n versus sum of n exponentials.....	25
Figure 2.5 Erlang- n versus sum of n exponentials.....	25
Figure 2.6 Erlang- n versus sum of n exponentials.....	26
Figure 2.7 95th percentile of Erlang- n and using weight function	28
Figure 2.8 99th percentile of Erlang- n and using weight function	29
Figure 2.9 Hypoexponential- n versus sum of n exponentials.....	31
Figure 2.10 Hypoexponential- n versus sum of n exponentials.....	31
Figure 2.11 95th percentile of hypoexponential-2 and equation 12	33
Figure 2.12 95th percentile of hypoexponential-2 and equation 12	34
Figure 2.13 95th percentile of hypoexponential-6 and equation 12	34
Figure 2.14 95th percentile of hypoexponential-6 and equation 12	35
Figure 2.15 95th percentile of hypoexponential-6 and equation 12	35
Figure 2.16 Coxian- n distribution.....	37
Figure 2.17 Coxian-2 distribution.....	37
Figure 2.18 A series of Coxian 2 distributions	38
Figure 2.19 95th percentile of phase type and equation 14	40
Figure 2.20 95th percentile of phase type and equation 14	40
Figure 2.21 7 component network model	42
Figure 2.22 Network model	45

Figure 2.23 AG latency data	46
Figure 2.24 SI latency data	47
Figure 2.25 TWSS latency data	47
Figure 2.26 The network under study	49
Figure 2.27 The minimum spanning tree using the average delay	49
Figure 2.28 The minimum spanning tree using the percentile delay	50
Figure 3.1 The tandem queueing network under study.....	55
Figure 3.2 The phase type distribution of the end-to-end delay	58
Figure 3.3 The CDF of D , D_1 , and D'	60
Figure 3.4 The upper and lower bounds of the bandwidth	62
Figure 3.5 Bandwidth results for a tandem network with 10 queues	64
Figure 3.6 Interpolation function vs simulation for various delay percentiles	66
Figure 3.7 Interpolation function vs simulation for various values of the delay T	67
Figure 3.8 Interpolation function vs simulation for various c_2 values	67
Figure 3.9 Interpolation function vs simulation for various ρ values	68
Figure 3.10 Bandwidth results for a tandem network with 10 queues for a single IPP.....	70
Figure 4.1 The tandem queueing network under study.....	76
Figure 4.2 The upper and lower delay bounds.....	81
Figure 4.3 95 th Percentile Delay results for a Tandem Network with 10 queues	84
Figure 4.4 Telepresence Trace. Packet Length Distribution (left), Arrival Bit Rate (right)...	86
Figure 4.5 Telepresence Trace. The 95 th Percentile of the T vs. Utilization (left), Bandwidth Required vs. T^D (right)	87
Figure 4.6 IPTV Trace. Packet Length Distribution (left), Arrival Bit Rate (right).....	88

Figure 4.7 IPTV Trace. The 95 th Percentile of the T vs. Utilization (left), Bandwidth Required vs. T^D (right)	89
Figure 4.8 WebEx Trace. Packet Length Distribution (left), Arrival Bit Rate (right).....	89
Figure 4.9 WebEx Trace. The 95 th Percentile of the T vs. Utilization (left), Bandwidth Required vs. T^D (right)	90
Figure 5.1 Tandem Queueing Network under study	96
Figure 5.2 Infinite Looping of Trace to create Background Traffic	101
Figure 5.3 Burstiness (left) and Autocorrelation (right) of combined Telepresence streams	102
Figure 5.4 Bandwidth Requirement for Fixed 95th Percentile Delay of 30msec (left). Difference between No Statistical Gain and Required BW (right).....	103
Figure 5.5 Bandwidth Requirement for Fixed 95th Percentile Delay of 50msec (left). Difference between No Statistical Gain and Required BW (right).....	104
Figure 5.6 Bandwidth Requirement for Fixed 95th Percentile Delay of 70msec (left). Difference between No Statistical Gain and Required BW (right).....	104
Figure 5.7 Burstiness (left) and Autocorrelation (right) of combined WebEx streams	106
Figure 5.8 Bandwidth Requirement for Fixed 95th Percentile Delay of 30msec (left). Difference between No statistical Gain and Required BW (right)	107
Figure 5.9 Bandwidth Requirement for Fixed 95th Percentile Delay of 50msec (left). Difference between No statistical Gain and Required BW (right)	107
Figure 5.10 Bandwidth Requirement for Fixed 95th Percentile Delay of 70msec (left). Difference between No statistical Gain and Required BW (right)	108
Figure 5.11 Burstiness (left) and Autocorrelation (right) of combined IPTV streams	109
Figure 5.12 Bandwidth Requirement for Fixed 95th Percentile Delay of 1 sec (left). Difference between No statistical Gain and Required BW (right)	110
Figure 5.13 Bandwidth Requirement for Fixed 95th Percentile Delay of 2 sec (left). Difference between No statistical Gain and Required BW (right)	111

Figure 5.14 Bandwidth Requirement for Fixed 95th Percentile Delay of 3 sec (left). Difference between No statistical Gain and Required BW (right)	111
Figure 5.15 Markov Chain of an IPP	112
Figure 5.16 Markov Chain of an (n+1) state MMPP representing n aggregate IPPs	113
Figure 5.17 Bandwidth Requirement for Fixed 95th Percentile Delay of 50 msec	113
Figure 5.18 Difference between No statistical Gain and Required BW	114
Figure 5.19 Pictorial Representation of big-theta Θ	116
Figure 5.20 The 95th percentile delay (left) and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3 when the background load on all three nodes increases proportionately ..	120
Figure 5.21 The 95th percentile delay (left) and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 3 while the background load on node 2 increases.	121
Figure 5.22 The 95th percentile delay (left), and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 2 while the background load on node 3 increases.	121
Figure 5.23 The 95th percentile delay (left), and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3 when the background load on all three nodes increases proportionately ..	122
Figure 5.24 The 95th percentile delay (left), and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 3 while the background load on node 2 increases.	123
Figure 5.25 The 95th percentile delay (left) and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 2 while the background load on node 3 increases.	123
Figure 5.26 Bandwidth and Jitter values for a 95th Percentile Delay of 50msec	124
Figure 5.27 Bandwidth and Percentile Delay values for Jitter of 10msec	125
Figure 5.28 Required bandwidth for dual bounds	126

Figure 5.29 Required bandwidth for dual bounds	127
Figure 5.30 Bandwidth and Jitter values for 95 Percentile Delay of 50msec	128
Figure 5.31 Bandwidth and Percentile Delay values for Average Jitter of 10msec	128
Figure 5.32 Required bandwidth for dual bounds	129
Figure 5.33 Required bandwidth for dual bounds	130
Figure 5.34 Bandwidth and Jitter values for 95 Percentile Delay of 50msec	130
Figure 5.35 Bandwidth and Percentile Delay values for Average Jitter of 10msec	131
Figure 5.36 Required bandwidth for dual bounds	131
Figure 5.37 Required bandwidth for dual bounds	132

Chapter 1

1 Introduction

1.1 End-to-End Quality of Service

Quality of Service (QoS) refers to the capability of a network to provide service differentiation for certain types of traffic. The primary goal of QoS is to provide specific traffic guarantees: controlled delay and jitter, bandwidth, and packet loss characteristics. Also it is important to make sure that providing priority to one type of traffic does not make the QoS of other traffic fail. Fundamentally, QoS can provide for better service to certain traffic by guaranteeing network resources that satisfy the user's demand. A key feature is that a network can provide different QoS levels to different services and each service has its own set of QoS requirements. For example, for an acceptable voice conversation over the Internet, it is recommended to have a one-way latency below 150 ms, a delay variation of about 30 ms, and a packet loss ratio below 1 percent.

Common QoS parameters include packet loss, end-to-end delay (latency), delay variation (jitter), availability (uptime), and data transfer rate (throughput). So, the question that arises is how to design a network so that to guarantee QoS. In the following section, we

review various schemes and concepts that have been standardized by Internet Engineering Task Force (IETF) so that to provide QoS in IP networks.

1.2 IP Network QoS Architectures

The IP-based Internet has not been designed to support QoS guarantees. This stems from the fact that the original Internet applications (e.g., email and FTP) are data oriented and they had little or no need for stringent guarantees. However, in the new era marked by the growing interest in providing voice and video services over IP networks, this situation is rapidly changing. This trend is paralleled by a phenomenal growth of the World Wide Web, with voice and video being further integrated into the design of Web pages. Given these recent trends, the IETF has been exploring various options for supporting multimedia traffic over IP.

1.2.1 Real Time Protocol (RTP)

RTP is the first step of supporting voice and video over Internet. RTP runs on top of UDP and interfaces it with the audio or video application. It is a **session layer protocol** and is transparent to network routers. Resultantly, it cannot be used to support network level QoS guarantees, still, it provides several functions useful for real time communications, including sequence numbering, timestamping, payload type identification etc. There are various ways of assigning RTP streams to media sources. Taking the example of teleconferencing, one may assign an RTP stream to each voice or video source in each direction. Alternatively, one RTP stream per voice and video bundle (as in case of MPEG) may also be assigned in each direction.

A sampled voice and/or video signal is encapsulated into an RTP packet, which is then encapsulated into a UDP packet. The receiver can detect lost RTP packets based on the sequence numbers of the received packets. The timestamp in the header of an RTP packet contains the relative sampling time of the video or voice signal. By comparing this time to the actual interarrival time, the network jitter can be computed. RTP relies on the real-time

control protocol (RTCP) to convey various types of information, including the number of transmitted packets and the number of lost packets. This information can be used by the sender to adjust the compression parameters and reduce the bit rate if necessary. In other words, RTP is best suited for adaptive video applications.

1.2.2 Integrated Services (IntServ)

Integrated Services Working Group of the IETF has developed a framework for Integrated Services (IntServ) that can support some form of QoS guarantees over the Internet in the 1990s. IntServ is a per-flow-based QoS framework with dynamic resource reservation. A specific flow, defined by the vector ‘IP source address, IP destination address, TCP/UDP source port, TCP/UDP destination port’, is recognized in the network and may deserve a specific treatment. Integrated services are based on an ATM-like paradigm, where each source–destination flow is distinguished. This implies the use of resource reservation, packet scheduling and buffer management, exactly as in ATM technology. It requires that routers through which a path traverses keep information about the status of the flow and that the status is periodically refreshed. IntServ uses the resource reservation protocol (RSVP) [1] to signal requests. The IntServ framework consists of two service offerings: Guaranteed Service [2] and Controlled Load Service [3]. The former guarantees a reliable upper bound on the metric ‘end-to-end packet delay’ based on worst-case assumption. The bound guarantee implies the use of control mechanisms as access control and traffic policing. By guaranteeing a certain bandwidth, the network guarantees the associated maximum packet delay (packet loss is guaranteed to be zero). The latter is aimed at providing a service similar to the level that a non-congested router would provide; it does not provide any upper bound but still requires packet scheduling and buffer management on per-flow basis.

IntServ uses two components to provide per flow guarantees: resource reservation and admission control. Resource reservation signals the network elements about the requirements (bandwidth and delay) of a particular flow. If the various network components (typically routers) are able to reserve the needed bandwidth, the signaling completes successfully. So, what does it mean for the router to "reserve" bandwidth? In effect, it means that the router

reserves the correct queueing preferences for the flow, such that the appropriate amount of bandwidth is allocated to the flow by the queueing tool.[4]. The second component, IntServ admission control, decides when a reservation request should be rejected. It is important because if all requests were accepted, eventually too much traffic would be introduced into the network, and none of the flows would get their requested service.

IntServ does not scale up well since the state of each path in each router is soft and as a result it needs to be continuously refreshed. Its signaling protocol RSVP, however, was reused successfully in MPLS.

1.2.3 Differentiated Services (Diffserv)

DiffServ [5] has been proposed by IETF in late 1990s with scalability as the main goal. In contrast to the IntServ model, Diffserv does not distinguish each traffic flow and resultantly it does not require a reservation protocol. It rather relies on the agreements between ISPs and clients to provide QoS guarantees. Classification of packets is performed based on the Differentiated Services (DS) field, which is part of the original ToS byte, in the header of an IP packet. Based on the field, the packets are classified into service classes, which are also referred as per hop behavior (PHB), when they enter the network. There are four main PHBs defined: assured forwarding (AF), expedited forwarding (EF), class selector (CS) and best effort (BE).

1.2.3.1 Assured Forwarding

The AF classes of PHB were designed to support data applications with assured bandwidth requirements. That is, packets will be forwarded with a high probability as long as the class rate submitted by the user does not exceed predefined contracted rate.

In the case of AF, each PHB contains two separate QoS functions: Queueing and Congestion Avoidance. As part of the first function, each router classifies the packets into four different classes, and packets from each class are placed in a separate queue. These queues are serviced using different queueing algorithms so as to provide the minimum configured bandwidth for each class. The second QoS function is Congestion Avoidance.

Typically, routers drop packets when a queue, in which the router needs to place the packet, is full. This action is called tail drop. Congestion Avoidance starts discarding packets before tail drop is required, hoping that the source will be notified early and eventually, fewer packets will be dropped,

1.2.3.2 Expedited Forwarding

The EF PHB is used to support applications that require low delay, low jitter, low packet loss, and assured bandwidth, such as VoIP.

The EF class suggests two QoS actions be performed to achieve the PHB. First, queueing must be used to minimize the time that EF packets spend in a queue. To reduce delay, the queueing tool should always service packets in this queue next. Also, anything that reduces delay reduces jitter. In addition to that, by always servicing the EF queue first, the scheduling discipline greatly reduces the queue length, which in turn greatly reduces the chance of tail drop due to the queue being full. Therefore, EF's goal of reducing delay, jitter, and loss can be achieved with a queueing method/scheduling discipline such as Priority Queueing. The second component of the EF PHB is policing. Policing implies that if the input load of EF packets exceeds a configured rate, the excess packets are discarded. Thus, EF protects other traffic types by capping the amount of bandwidth for the class.

1.2.3.3 Class Selector

CS PHB is used for backward compatibility with IP precedence ToS field. It uses the "bigger-is-better" logic, i.e., the bigger the value in DS field, the better the QoS treatment.

1.2.3.4 Best Effort

The default BE PHB has no committed resources and yet it cannot be starved by other PHBs. This is usually supported by allocating some nominal bandwidth to the BE flows.

Continuing our general discussion on DiffServ, the ingress router, in addition to marking the DS field, also applies traffic conditioning functions to control the amount of traffic entering the network. Traffic conditioning includes shaping (smoothing the rate at which packets are sent into the network) and policing (dropping packets that are in excess of a subscribed-to rate; or re-coloring the ones exceeding the rate, so that the probability of

dropping them increases when there is congestion in the core). A good example of a policer is the single rate three color marker scheme. The green, yellow and red actions have to do with dropping and marking of a packet. Typical action is as follows. Green action means that the packet is conformant and it should be let into the network. Yellow action means that the packet is not conformant, but it can be marked and let into the network. Finally, red action means that the packet is not conformant and it will be dropped. [6] Each node within the network then applies different queueing and dropping policies on every packet based on the marking that packet carries.

DiffServ is aimed at overcoming the scalability problem mentioned for the integrated services and does not require per-flow state and signaling. In practice, traffic is divided into classes that deserve a common service in the network. The advantage stands in the aggregation of many flows into a single traffic class, whose packets are forwarded in the same way in a router. This permits DiffServ to scale up to large size networks. However, the drawback is that no service per-flow can be guaranteed. So, while IntServ has inherent scalability problems, DiffServ does not provide explicit means for the application to negotiate a service level with the network. Also, unlike RSVP/IntServ, DiffServ needs to be provisioned. Setting up the various classes throughout the network requires knowledge of the applications and traffic statistics for aggregates of traffic on the network. This process of application discovery and profiling can be time-consuming. Still, DiffServ is widely used in the Internet, and it predates MPLS. Also some models using IntServ at the edge and DiffServ at the core of the network have been proposed [7]

1.2.4 Traffic Engineering

One of the biggest drawbacks of both the IntServ and DiffServ models is the fact that signaling/provisioning happens separately from the routing process. There may exist a path (other than the non-default Interior Gateway Protocol, such as OSPF, ISIS, EIGRP, and so on or Exterior Gateway Protocol, such as BGP path) in the network that has the required resources, even when RSVP/DiffServ fails to find the resources. This is where Traffic

Engineering (TE) and MPLS come into service. True QoS, with maximum network utilization, will arrive with the combination of traditional QoS and routing.

Traffic Engineering refers to the process of selecting the paths chosen by data traffic in order to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and traffic performance. The goal of TE is to compute path from one given node to another such that the path does not violate any constraints (bandwidth/administrative requirements) and is optimal with respect to some scalar metric. Once the path is computed, TE is responsible for establishing and maintaining forwarding state along such a path.

The lack of per-flow mechanisms in the DiffServ approach can be bypassed by proper traffic engineering techniques that assign a proper dimension to the bandwidth pipe assigned to each traffic class. Aref Meddeb, ISIT Com, in [8] showed that DiffServ offers bandwidth savings, relatively simple deployment, and tight SLA capabilities, especially when combined with TE. However, the benefit of DiffServ-aware TE does not seem to justify its complexity as surveyed by X. Xio in [9]. Furthermore, A. Mohammad et al [10] states that “DiffServ cannot commit to any specific value of delay or jitter without the help of other protocols or mechanisms such as bandwidth brokers or MPLS.”

1.2.5 Multiprotocol label switching (MPLS)

DiffServ can control the traffic submitted by a user so that it is conformant to the agreed upon SLA. Also, a scheduler and a queueing policy at each router makes sure that the traffic associated with each code point gets its appropriate quality of service. However, DiffServ cannot control the total traffic that goes through a link, as there is no explicit bandwidth allocation on per link basis. In view of this, this problem is typically solved by over-engineering the network. This is an expensive solution, and still it does not provide any guarantees in the presence of link failures. MPLS on the other hand can guarantee bandwidth per connection and it can also provide protection in the presence of link failures.

MPLS is a technology convergence between ATM and IP [11] [12]. MPLS sets up paths in a network along which packets that carry appropriate labels can be forwarded very efficiently (i.e. the forwarding engine does not use at the IP address, rather it uses the label to forward the packet). Not only does this allow packets to be forwarded more quickly, it also allows paths to be set up in a variety of ways: the path could represent the normal destination-based routing path, it could represent a policy-based explicit route, or it could represent a reservation-based flow path. Ingress routers classify incoming packets and wrap them in an MPLS header that carries the appropriate label for forwarding by the interior routers. In the MPLS model, the labels are distributed by the RSVP-TE, which is an extension of RSVP, and which sets up effectively a label switched path (LSP) along the label switched routers (LSR). RSVE-TE can be used to set up LSPs using the forwarding table (e.g. OSPF) or explicit routes based on maximizing one or more QoS parameters as well as taking into account policy-based rules. It is important to note that this label state can be per flow (as it was with IntServ) but usually represents some aggregate (e.g. between some source–destination pair). Therefore, the state produced by MPLS is manageable and scalable [13]

1.2.6 MPLS Overlay Networks

Although MPLS label switching provides the underlying technologies in forwarding packets through MPLS networks, it does not provide all the components for Traffic Engineering support such as traffic engineering policy. The existing Interior Gateway Protocols (IGP) are not adequate for traffic engineering. Routing decisions are mostly based on shortest path algorithms that generally use additive metric and do not take into account bandwidth availability or traffic characteristics.

As discussed by C. S. Paper in [14], the easiest way to provide such features would be to use an overlay model, which enables virtual topologies on top of the physical networks. The virtual topology is constructed from virtual links that appear as physical links to the routing protocol. Further, the overlay model should be able to provide: (1) constraint based routing, (2) traffic shaping and traffic policing functionality, (3) survivability of the virtual links etc.

These capabilities allow easy movement of traffic from an over subscribed link to an underused one. MPLS is the overlay model used by Traffic Engineering

1.2.7 Queue Management and Scheduling Policies

In the interior routers, differentiated packets have to be handled differently. To do so, the router may employ multiple queues, along with some class-based queueing (CBQ) service discipline or simple priority queueing. Generally, delay-sensitive traffic will be serviced sooner, and loss-sensitive traffic will be given larger buffers. The loss behaviour can also be controlled using various forms of random early detection (RED). These disciplines using probabilistic methods to start dropping packets when certain queue thresholds are crossed, in order to increase the probability that higher quality packets can be buffered at the expense of more dispensable packets. Some of the widely used scheduling policies are described in the table below

Table 1.1 Scheduling Policies

Scheduling Policy	Queue Service Algorithm
Priority Queueing (PQ)	Strict service; always serves higher-priority queue over lower queue
Custom Queueing (CQ)	Serves a configured number of bytes per queue, per round-robin pass through the queues. Result: Rough percentage of the bandwidth given to each queue under load
Weighted Fair Queueing (WFQ)	Each flow uses a different queue. Queues with lower volume and higher IP precedence get more service; high volume, low precedence flows get less service.
Class-Based Weighted Fair Queueing (CBWFQ)	Results in set percentage bandwidth for each queue under load.
Low Latency Queueing (LLQ)	LLQ is a variant of CBWFQ, which makes some queues "priority" queues, always getting served next if a packet is waiting in that queue. It also polices traffic.
Modified Deficit Round-Robin (MDRR)	Similar to CQ, but each queue gets an exact percentage of bandwidth. Supports LLQ mechanism as well.

1.3 End-to-end Bandwidth Guarantees

As discussed in the previous section, MPLS is able to provide end-to-end bandwidth guarantee whereas DiffServ is generally unable to do so. This is mainly because though an ISP can have a general idea of the traffic entering his network, he does not have any control on the path followed by the traffic or the domains it pass through. End-to-end quality of service usually requires a method of coordinating resource allocation between one autonomous system and another. IntServ can also provide bandwidth guarantees but it is not used much due to its scalability issues.

From this point onwards, we will consider an MPLS network and we will focus on the problem of calculating bandwidth for an LSP to ensure a quality experience for the user when certain characteristics about the arrival traffic are known. It is interesting to note that the term “quality experience” has different meanings for different applications. For example for real time voice, the primary concern will be to minimize end-to-end delay but it is resilient to some packet loss. Whereas a data centric application like FTP requires zero packet loss but can endure network delays. There is a spectrum of applications with different quality requirements and expectations from the network. Generally, there are two ways of doing bandwidth allocation; one is to guarantee packet loss and the other to bound end-to-end delay and jitter. Below we give a brief survey of techniques for guaranteeing packet loss. The subject matter of this thesis is bandwidth allocation under end-to-end delay guarantees and a detailed review of the relevant literature is given in the following section 1.3.2.

1.3.1 Bandwidth allocation based on Packet Loss

Statistical multiplexing is a mechanism for reducing bandwidth requirement especially for bursty traffic sources. It has been used over the Internet to improve network utilization, but without providing any performance guarantees. Statistical Multiplexing is a spatial aggregation mechanism by which several individual streams are asynchronously superposed and transported over the same channel. As a result of superposition, the resultant traffic exhibits less variability (smoother bit rate behavior) than the original streams. Thus the

bandwidth allocation to the aggregate traffic results in a reduction in the per stream allocated bandwidth, where the reduction is proportional to the burstiness of the multiplexed sources.

Several queueing theoretic papers have analyzed the loss probability in finite buffers or the queueing tail probability in infinite buffers. For instance, Kim and Shroff model the input traffic as a general Gaussian process and derive an approximate expression for the loss probability in a finite buffer system [15].

An early experimental study by Villamizar and Song recommended that the buffer size should be equal to the bandwidthdelay product (BDP) of that link [16]. The “delay” here refers to the RTT of a single and persistent TCP flow that attempts to saturate that link, while the “bandwidth” term refers to the capacity C of the link. That rule requires the bottleneck link to have enough buffer space so that the link can stay fully utilized while the TCP flow recovers from a loss-induced window reduction.

The BDP rule results in a very large buffer requirement for high-capacity long-distance links. At the same time, such links are rarely saturated by a single TCP flow. Appenzeller et al. concluded that the buffer requirement at a link decreases with the square root of the number N of “large” TCP flows that go through that link [17]. According to their analysis, the buffer requirement to achieve almost full utilization is $B=(CT)/\sqrt{N}$, where T is the average RTT of the N (persistent) competing connections. The key insight behind this model is that when the number of competing flows is sufficiently large, which is usually the case in core links, the N flows can be considered independent and nonsynchronized, and so the standard deviation of the aggregate offered load (and of the queue occupancy) decreases with $\sqrt{N}$. An important point about this model is that it aims to keep the utilization close to 100% without considering the resulting loss rate.

Dhamdhere et al. considered the buffer requirement of a drop-tail queue given constraints on the minimum utilization, the maximum loss-rate, and, when feasible, the maximum queueing delay [18]. They derive the minimum buffer size required to keep the

link fully utilized by a set of N heterogeneous TCP flows while keeping the loss rate and queueing delay bounded.

Recently, the ACM Computer Communications Review (CCR) has hosted a debate on buffer sizing through a sequence of letters [19], [20], [21], [22]. On one side, Enachescu et al in [20] and Raina et al in [21] have proposed significant reduction in the buffer requirement based in results from earlier studies [17]. They argue that 100% link utilization can be attained with much smaller buffers, while large buffers cause increased delay, induce synchronization, and are not feasible in all-optical routers. On the other side of the debate, Dhamdhere and Dovrolis in [19] and Vu-Brugier et al in [22] highlight the adverse impact of small buffer size in terms of high loss rate and low per-flow throughput. Dhamdhere and Dovrolis argued that the recent proposals for much smaller buffer sizes can cause significant losses and performance degradation at the application layer [19]. Similar concerns are raised by Vu-Brugier et al. in [22]. That letter also reports measurements from operational links in which the buffer size was significantly reduced.

Lakshmikantha et al. [23] have showed that depending on the ratio between the “edge” and “core” link capacities, the buffer requirement can change from $O(1)$ (just few packets) to $O(CT)$ (in the order of the BDP).

1.3.2 Bandwidth allocation based on end-to-end delay

A well known approach of allocating bandwidth so as to guarantee end-to-end delay is called equivalent bandwidth, proposed originally for ATM networks, see Perros [6]. In addition, a survey of various call admission algorithms can be found in Wright [24].

Charara et al [25] compare three methods for the evaluation of end-to-end delays for avionics network architectures. The new generation of avionics embedded systems are marked by the characteristics of increased number of integrated functions, increased number of connections between these functions and hence the growth of the number of multipoint communication. Airbus, for the new A 380 generation, has adopted the Switched Ethernet technology with bi-directional links (AFDX, 802.1D) and static routing. The authors are

interested in finding out the worst case end-to-end delay of the system. The end-to-end traffic is characterized using Virtual Links (VL) where a VL is defined as a logical unidirectional connection from one source port to one or more destination ports. Hence, VL is a path with multicast characteristics. The first approach used to evaluate the upper bound end-to-end delay is that of network calculus. Given an elementary entity that offers service curve β to an input flow constrained by an arrival curve α , the calculus brings the arrival curve α^* of the output flow: $\alpha^* = \alpha \otimes \beta$ where $\alpha \otimes \beta$ defined by: $\alpha \otimes \beta(t) = \sup_{u \geq 0} \{(\alpha(t+u) - \beta(u))\}$. Using a Network Calculus tool to propagate these results on the complete network, the approach analytically derives sure upper bounds on delays. However, the bounds are extremely pessimistic as observed when crossed checked by the experimental upper bounds obtained by simulation of a set of scenarios. The ratio of the end-to-end delay obtained by simulation and the one calculated with the network calculus is mostly between 5% and 40% with the ratio 70% achieved only when all VL paths have a length 1. However, simulation model may miss rare events hence undermining the worst case delay and so a more comprehensive approach of modeling the network as timed automata with model checking may be used. It gives an exact worst case end-to-end delay by exploring all the possible states of the system. However the number of states is directly dependent on the size of the output queues per switch and the external arrivals and may lead to combinatorial explosion.

Koij et al [26] improve on the percentile upper bound on the end-to-end delay as experienced by various real time CBR sources inside a homogenous network. The node model consists of two queues (one for real time and one for low priority traffic), served by a non-pre-emptive head of line scheduler. Though the real time packets get absolute priority over best effort data flows, the data packets do cause some interference due to the non-preemptive nature of the scheduler. Assuming that the two queues are independent and the packet lengths are constant, the γ -percentile of the upper bound on the end-to-end delay of a K node network is calculated as the arithmetic sum of the γ -percentile of the K times convolution of the delay of a single M/D/1 node and K times the service time of a single data packet. The authors model a single node as an M/G/1 non-preemptive queueing system and calculate the system waiting time using Pollaczek-Khinchin formula. This is then convoluted

K times to get the end-to-end delay value on which a γ -percentile may be constructed. Experimental evidence shows that in access networks (low bit rates) the improvement over the upper bound, due to the later method, can be up to 45% and in the order of tens of milliseconds, whereas in core networks (high bit rates) the upper bound is reasonably accurate in predicting the end-to-end delay.

Vleeschauwer et al [27] have described four different approximations to compute the γ -percentile of the total queueing delay in a heterogeneous network where each node can be represented by an M/G/1 queue. The queues are assumed to be independent, but not necessarily identical. The simplest approximation is based on the assumption that the distribution of the total queueing delay, consisting of N statistically independent queueing delays, shapes towards a Gaussian distribution. Hence the percentiles can be calculated as: $\mu + \text{erfc}^{-1}(P)\sigma$ where erfc is the complimentary error function of the Gaussian distribution. A heuristic formula is subsequently developed along the same lines but this time the weighing factor instead of being erfc is chosen such that , the formula would be exact if the individual delays were exponentially distributed. The third approach is based upon the dominant pole associated with each M/G/1 node. If the moment generating function of the delay in one node can be written as: $D_n(s)=H_n(s)/(s-p_n)$, where $(s-p_n)$ is the dominant pole, the end-to-end distribution is a weighted sum of the CDFs of Erlang variables. Experimental evidence shows that the second and third approach outperforms Gaussian approximation in all cases, the main reason being that a very large number of nodes (order of a few hundred) is required to make the resultant CDF a Gaussian distribution. Whereas the errors for the two methods are smaller than 1% as soon as the load on at least one of the nodes is high enough (0.7 for second method and 0.5 for the third method). The last method discussed involves the numerical inversion of the Laplace transforms following Abate and Whitt [28]. This method is the most complicated one but it works very well but not for very high percentiles because of the discretization and truncation errors which are inherent in the method.

Goyal et al [29] present a way to determine an upper bound on the end-to-end packet delays of a flow that passes through a network that employs Guaranteed Rate scheduling

algorithms. The authors define Guaranteed Rate algorithms (GR) as a class of schedulers that can guarantee a deadline by which a packet of an accepted flow will be transmitted, e.g., virtual clock, self clocked fair queueing, generalized processor sharing. Using the arrival time of the packet, transmission deadline calculated by the first scheduler, packet length and the flow's associated rate per node, a mathematical upper bound is constructed that is shown to work well with the sources conforming to Leaky Bucket and Exponentially Bounded Burstiness.

Lehoczky and Yeung [30] work on a new queueing theory methodology called real-time queueing theory, that allows one to keep track of the deadlines associated with each of the tasks/packets in the system. An important performance metric in this theory is the packet lateness probability i.e., the fraction of packets which miss their deadline. The theory can be applied only when the traffic intensity on the server approaches 1. In this case, under very general circumstances (general i.i.d interarrival time and service time distributions), the occupancy of the queueing network can be treated as a reflected Brownian network process with drift whose equilibrium probability distribution is of product form if the first two moments of the interarrival time and service time distributions satisfy certain conditions. Considering a simple two stage distributed network and using the product form solution, the authors computed a closed form expression of the PDF of the end-to-end delay further from which the proportion of late packets of each session is determined. The authors provide closed form expressions for the proportion of late packets for two scheduling disciplines EDF and FIFO and for constant and uniform distributed deadlines. Simulations illustrate good accuracy of the closed form expressions.

Iftikhar et al [31] present an analytical model based on G/M/1 queueing system for modeling of wireless IP traffic behavior under the assumption of four different classes of self-similar traffic per single router. The model implements four queues based on G/M/1 queueing system and was analyzed on the basis of priority with no preemption. Major contribution of the paper is to calculate packet interarrival time distribution for a particular self-similar traffic. Also distribution of cross interarrival time between different types of

packets is derived on the basis of single packet results. Second major contribution is the derivation of explicit expressions for expected waiting time in the queue for each of the priority classes. The paper also formulates the embedded Markov chain of the G/M/1 node by considering all possible states and derives the corresponding transition probabilities.

An exact numerical expression of the end-to-end delay in a tandem queueing network can also be obtained by calculating its Laplace transform and then inverting it numerically to obtain delay percentiles. This approach was used by Xiong and Perros, [32] within the context of resource optimization of web services using a tandem queueing network with Poisson arrivals and exponentially distributed service times.

Yeung and Lehoczky [33] used a Brownian process to study a two-stage queueing network with customers having deadlines (constant and uniformly distributed) and calculated bounds for two different scheduling disciplines, early-deadline first and FIFO. Fractional Brownian Motion (FBM) was used by Lelarge et al [34] to show that the end-to-end delay of a tagged flow in a tandem queueing network, and more generally in a tree network, is completely dominated by the queue with the maximal Hurst parameter.

1.4 Summary of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, we define percentiles and discuss their statistical significance in various networking situations. We then present an exact analytic expression for adding percentiles of random variables whose PDF is a mixture of exponentials. Specifically we have considered exponential, Erlang, hypoexponential, hyperexponential, and Coxian-2 distributions. We also discuss the applicability of percentile calculations to routing protocols, presenting Dijkstra's algorithm as an example.

In Chapter 3 we use end-to-end delay percentiles, as the Quality of Service restriction, to propose a bandwidth allocation scheme for an MPLS-enabled IP network. In other words, we are concerned with the calculation of the bandwidth to be allocated at the output port of each router along the path of a connection, so that the end-to-end delay D is

less than or equal to a given target delay value T with a probability γ , i.e., $P(D \leq T) = \gamma$. We assume that the arrival of packets follow an MMPP2 process, because it is capable of capturing burstiness and autocorrelation characteristics commonly present in network traffic while satisfying a reduced complexity. We first construct an upper and a lower bound on a given percentile of D , from which we obtain bounds of the bandwidth such that $P(D \leq T) = \gamma$, for given T and γ . These two bounds are then combined using an interpolation function to obtain an accurate estimate of the bandwidth. The main contribution of the scheme is that the upper and lower bounds are constructed by analyzing only the first queue of the tandem queueing network.

In Chapter 4, we extend the approach described in Chapter 3 to the case where the arrival process is a two-state Markov arrival processes (MAP). A MAP can represent a variety of processes that includes, as special cases, the Poisson process, the phase-type renewal processes, the MMPP and superposition of these. Later in the chapter, we also used a MAP2 to successfully approximate the packet arrival process of various video streams, such as Cisco's Telepresence, IPTV and WebEx.

We take the bandwidth allocation of video using traces a step further in Chapter 5. The algorithm described in Chapter 4 for calculating a percentile of the end-to-end delay of a video stream, cannot handle the presence of background traffic that competes with the video within the output port of a router. In view of this, we propose an exact and efficient algorithm for calculating any percentile of the end-to-end delay in the presence of background video traffic, assuming that all video flows are characterized by packet traces and an arriving packet keeps its packet length in the entire queueing network. Using the algorithm, we calculate the required bandwidth as a function of the number of homogeneous video streams so that the end-to-end percentile delay remains the same. Also we present a brief analysis on jitter and percentile of a tagged stream in the presence of background traffic of varying intensities. Later in the chapter we extend the analysis to calculate how the bandwidth requirements of a flow changes as the background traffic varies on all the nodes constituting the end-to-end path. Finally we demonstrate the applicability of the algorithm for dual

bounds situations, i.e., how can we use the algorithm to calculate the bandwidth required for a flow that needs to satisfy constraints on both percentile delay and average jitter at the same time.

Chapter 2

2 Percentile Arithmetic

2.1 Introduction

Let us consider a performance metric such as the response time of a router or a web service or a software process. Typically, we use the average of this metric as an indicator of its performance. For instance, we will say that the average time it takes for a specific web service is 2 ms. However, we all know that averages can be misleading as they do not represent the range of values of the response time under study. On the other hand, a percentile of the metric provides a better understanding than the mean value, since it bounds statistically the behavior of a system. The c percentile, such as 95th percentile, of a variable X is a value below which X lies $c\%$ of the time.

There is a plethora of situations in practice where knowledge of the percentile calculus of a random variable could significantly improve the efficiency of a process. Below, we describe some examples where this problem arises.

Response time in a web service: The execution of a web-based service may involve several sites, each carrying out part of the service flow. Given that each site can guarantee the 95th percentile of its own response time, the question is what is the 95th percentile of the total end-to-end time. This end-to-end percentile can then be used in the negotiation of the contract with the user.

Testing a large suite of software: Let us consider a suite of software components that provide a service. Such a suite can be the set of software designed to provide IMS services. Testing for bottlenecks is standard routine before the software is released. However, due to the complexity of IMS, it is impossible to have all the components present in a lab. In view of this, the components that are not available for testing are often represented by idealized simulations which are generally built as "no-op" stubs and return results artificially fast. As a result, the end-to-end response distribution cannot be reliably obtained. An alternative solution, is to test only sub-groups of software components at a time and obtain the percentile of the response time for each group. The individual results can then be aggregated to get an estimate of the end-to-end percentile response time. Again, we see here the importance of being able to add up percentiles of response times.

QoS in multi-domain routing: User traffic typically originates at a local area network, and then it traverses an access network before it is channeled into a WAN or a series of WANs, each operated by a different ISP, to reach its destination which may be another access network. Time sensitive traffic, such as VoIP and interactive video, needs to be treated by the ISPs in such a way so that the end-to-end delay and the end-to-end jitter is minimum. Again the same problem arises here. Each ISP typically guarantees the 95th percentile of the time to traverse its domain and of the jitter generated within the domain due to congestion. Based on the individual percentiles, what guarantees can we provide for the end-to-end delay and jitter?

Controlling the power budget: An interesting problem arises in green data centers. Specifically, let us assume that we want to keep power consumption so that 95% of the time it is less than a total power budget P . The question arises as to how this can be calculated if

we know the percentiles of the power consumption of the individual devices or groups of devices.

A similar problem is also encountered in cloud computing where multiple software components run in a virtual environment on the same blade, one component per virtual machine (VM). Each VM is allocated a virtual CPU, which is a fraction of the blade's CPU. The hypervisor automatically monitors CPU usage for each VM. The question here is how do we allocate the blade's CPU to the multiple VMs running on the same blade so that a given percentile of the response time of each VM is satisfied while at the same time the percentile of the overall power consumption is bounded.

Despite the many uses of the percentile of a random variable, very little is known as to how to add and divide percentiles. In this Chapter, we present an exact analytic expression for adding percentiles of random variables whose PDF is a mixture of exponentials. Specifically we have considered exponential, Erlang, hypoexponential, hyperexponential, and Coxian-2 distributions.

The Chapter is organized as follows. In section 2.2 we give a literature review and define the problem. In section 2.3, we present an expression for adding up percentiles of random variables which are exponentially distributed with the same or different rate. The results are extended to Coxian-2s. Two real-life examples are presented in sections 2.4 and 2.5. In section 2.6 we discuss the applicability of percentile calculations to routing protocols, presenting Dijkstra's algorithm as an example. Finally, the conclusions and future research directions are given in section 2.7.

2.2 Problem Statement

As mentioned above, very little work has been done about how to add percentiles and also how to partition a percentile to individual components. Kreifeldt and Nah [35] reported on the error of adding and subtracting percentiles of anthropometric dimensions in order to derive other relevant dimensions. The work focuses particularly on Gaussian distributions

and adding/subtracting equal percentile points. The key findings are that the error between the assumed (added/subtracted) percentile and the actual percentile depends on the percentile point, the correlation coefficients and the standard deviation ratios of the components. Also the error decreases as the correlation increases and/or the standard deviation ratio decreases. The issue of adding percentiles was also addressed in a white paper on inter-provider QoS by the MIT Communications Futures Program [36]. The paper addressed the issue of how to allocate the end-to-end response time, packet loss and jitter across multiple operators. The response time was expressed as the mean, and the jitter was expressed as a percentile of the inter-arrival time at the destination. The authors proposed a method for adding the individual operators' jitter. As will be shown in section 2.5 their method is grossly inaccurate.

The problem studied in this Chapter can be defined in general terms as follows. Let us consider a system consisting of n individual and independent components as shown in Figure 2.1.

Figure 2.1 Composition of n components

We will assume that for each component we know the percentile of a metric of interest, such as the response time, power consumption, and jitter. We will then calculate the percentile of the end-to-end metric over all the n components.

2.3 Calculation of the Weight Function

2.3.1 Exponential Components with identical rate parameter

We start by assuming our metric of interest is exponentially distributed with a probability density function (PDF): $f(x) = \mu e^{-\mu x}$ and a cumulative density function (CDF):

$F(X) = 1 - e^{-\mu x}$. The sequence of the n components can be represented by a sequence of n exponential distributions as shown in Figure 2.2.

Figure 2.2 A sequence of exponential distribution

In the case where $\mu_1 = \mu_2 = \dots = \mu_n = \mu$, the end-to-end distribution is an Erlang distribution. Its PDF $f(x)$, CDF $F(X)$ and Laplace transform $f_x^*(s)$ are as follows:

$$f_x^*(s) = \left(\frac{\mu}{\mu + s} \right)^n$$

$$f(x) = \mu e^{-\mu x} \frac{(\mu x)^{n-1}}{(n-1)!}$$

$$F(X) = 1 - \sum_{i=0}^{n-1} e^{-\mu x} \frac{(\mu x)^i}{i!}$$

Using the above CDF formula of the end-to-end distribution, we can compute a specific percentile, say c . We have that

$$1 - \sum_{i=0}^{n-1} e^{-\mu x} \frac{(\mu x)^i}{i!} = c$$

from which we can solve for x . The value of x is the c^{th} percentile of the distribution. That is c percent of the area under the curve $f(x)$ lies before x .

Of interest is to examine how the c^{th} percentile of the individual components are related to the c^{th} percentile of the end-to-end distribution. As a first step, we plotted the c^{th} percentile of the Erlang distribution against the sum of the c^{th} percentiles of the individual components. In Figure 2.3 we plotted the 80th percentile of an Erlang- n along with the sum of

the 80th percentile of the individual exponential components, with $\mu=5.0$ and 1.0. The plots are given as a function of the number of components n , which was varied from 1 to 30.

Figure 2.3 Erlang-n versus sum of n exponentials

Similar results are given in Figure 2.4 and Figure 2.5, but for the 95th and 99th percentile respectively. The value of μ was set to 1.

Figure 2.4 Erlang-n versus sum of n exponentials

Figure 2.5 Erlang-n versus sum of n exponentials

Finally Figure 2.6, is similar to Figure 2.5, but the number of components was raised from 1 to 100. This was done to see if there are any hidden data patterns as the number of

components increases. However, the difference between sum and percentile of Erlang-n is still found to be exponentially increasing.

Figure 2.6 Erlang-n versus sum of n exponentials

In general, for a given value of μ , the arithmetic sum of the individual c^{th} percentiles is always greater than the c^{th} percentile of the end-to-end distribution, and thus the difference increases as n increases. Also the difference increases as c increases. Finally for a given value of c , the growth of difference between the arithmetic sum of the individual c^{th} percentiles and the c^{th} percentile of the end-to-end distribution for each n is independent of the value of the parameter μ . This observation is substantiated below.

Let x_{exp} be the $c_{\text{exp}}^{\text{th}}$ percentile of an exponential distribution with rate μ i.e., of a component, and let x_{Erl} be the $c_{\text{Erl}}^{\text{th}}$ percentile of an Erlang distribution with n stages, each with parameter μ , i.e., of the end-to-end delay. Then, we have that

$$1 - e^{-\mu x_{\text{exp}}} = c_{\text{exp}} \quad (1)$$

and

$$1 - \sum_{i=0}^{n-1} e^{-\mu x_{Erl}} \frac{(\mu x_{Erl})^i}{i!} = c_{Erl} \quad (2)$$

Given the values of c_{exp} , c_{Erl} and x_{exp} , we are interested in finding out a weight function w such that:

$$x_{Erl} = w x_{exp} \quad (3)$$

Equation 1 can be re-written as:

$$x_{exp} = \frac{-\ln(1 - c_{exp})}{\mu}$$

Hence equation 3 becomes:

$$x_{Erl} = w \frac{-\ln(1 - c_{exp})}{\mu}$$

Putting the value of x_{Erl} in equation 2, hence equation 2 becomes:

$$1 - \sum_{i=0}^{n-1} e^{-\mu w \frac{-\ln(1 - c_{exp})}{\mu}} \frac{(\mu w \frac{-\ln(1 - c_{exp})}{\mu})^i}{i!} = c_{Erl}$$

or

$$1 - c_{Erl} = e^{w \ln(1 - c_{exp})} \sum_{i=0}^{n-1} \frac{(-w \ln(1 - c_{exp}))^i}{i!}$$

or

$$\ln(1 - c_{Erl}) = w \ln(1 - c_{exp}) + \ln \sum_{i=0}^{n-1} \left\{ \frac{(-w \ln(1 - c_{exp}))^i}{i!} \right\} \quad (4)$$

The expression has no analytical closed-form solution, however it can be solved numerically for a given c and n (e.g. by using Bisection or Newton's method in Matlab). This is our required weight function and when multiplied by x_{exp} , gives us the required value of x_{Erl} .

It should be noted here that c_{exp} need not be the same as c_{Erl} , i.e., equation 4 calculates a weight for converting exponential percentile to any Erlang percentile (and vice versa). Also μ is not present in equation 4, confirming our previous observation that the difference between the arithmetic sum of individual c^{th} percentiles and the c^{th} percentile of the end-to-end distribution for each n is independent of the value of the parameter μ .

Thus given a fixed parameter μ , equation 3 and 4 give us an exact formula to calculate any percentile of an Erlang distribution (c_{Erl}), given a specific percentile of the corresponding exponential distribution (c_{exp}). Figure 2.7 and Figure 2.8 give results for the 95th and 99th percentile respectively. In each figure we plotted x computed using equation 3 and 4 and also using the CDF of the Erlang- n , for $n = 1, 2, 3, \dots, 30$. As expected, the results are identical.

Figure 2.7 95th percentile of Erlang-n and using weight function

Figure 2.8 99th percentile of Erlang-n and using weight function

2.3.2 Exponential Components with different rate parameter

In the case where the rate parameters of the components are not necessarily the same, the end-to-end distribution is a hypoexponential distribution. This can be seen as a generalized Erlang distribution where each stage i has a different rate μ_i . In general, if we have n independently distributed exponential random variables X_i , then the random variable, $X = \sum_{i=1}^n X_i$ is hypoexponentially distributed. The hypoexponential has a minimum squared coefficient of variation of $1/n$, and its Laplace transform is:

$$f_x^*(s) = \frac{\mu_1}{\mu_1 + s} \cdot \frac{\mu_2}{\mu_2 + s} \cdots \frac{\mu_n}{\mu_n + s}$$

We note that the PDF and CDF formulas of the hypoexponential distribution are not readily available in the literature. These can be obtained by inverting its Laplace transform as follows. Using partial fraction expansion, we have that:

$$f_x^*(s) = \frac{\mu_1 \mu_2 \cdots \mu_n}{(\mu_2 - \mu_1)(\mu_3 - \mu_1) \cdots (\mu_n - \mu_1)} \frac{1}{\mu_1 + s} + \frac{\mu_1 \mu_2 \cdots \mu_n}{(\mu_1 - \mu_2)(\mu_3 - \mu_2) \cdots (\mu_n - \mu_2)} \frac{1}{\mu_2 + s} + \cdots + \frac{\mu_1 \mu_2 \cdots \mu_n}{(\mu_1 - \mu_n)(\mu_2 - \mu_n) \cdots (\mu_{n-1} - \mu_n)} \frac{1}{\mu_n + s}$$

We observe that $\frac{\mu_i}{\mu_i + s}$ is the Laplace transform of an exponential distribution with parameter μ_i .

Hence we have,

$$f(x) = \frac{\mu_1 \mu_2 \dots \mu_n}{(\mu_2 - \mu_1)(\mu_3 - \mu_1) \dots (\mu_n - \mu_1)} e^{-\mu_1 x} +$$

$$\frac{\mu_1 \mu_2 \dots \mu_n}{(\mu_1 - \mu_2)(\mu_3 - \mu_2) \dots (\mu_n - \mu_2)} e^{-\mu_2 x} + \dots + \frac{\mu_1 \mu_2 \dots \mu_n}{(\mu_1 - \mu_n)(\mu_2 - \mu_n) \dots (\mu_{n-1} - \mu_n)} e^{-\mu_n x}$$

or

$$f(x) = \sum_{i=1}^n \left\{ \mu_i e^{-\mu_i x} \prod_{\substack{j=1 \\ j \neq i}}^n \frac{\mu_j}{\mu_j - \mu_i} \right\} \quad (5)$$

From equation 5 we can obtain the CDF of a hypoexponential distribution. We have

$$F(X) = \sum_{i=1}^n \left\{ (1 - e^{-\mu_i x}) \prod_{\substack{j=1 \\ j \neq i}}^n \frac{\mu_j}{\mu_j - \mu_i} \right\} \quad (6)$$

The observations made above for the Erlang distribution are also valid for the hypoexponential distribution. In Figure 2.9 we plotted the sum of the 95th percentile of n exponential components, where parameter value of each component is , $\mu_i = i$, $i=1,2,3,\dots,n$ along with the 95th percentile of the end-to-end hypoexponential distribution. Figure 2.10 gives a similar plot for the 99th percentile.

Figure 2.9 Hypoexponential-n versus sum of n exponentials

Figure 2.10 Hypoexponential-n versus sum of n exponentials

It should be mentioned here that the shape of the c^{th} percentile of the hypoexponential distribution depends on the value of the μ parameters. The graphs of Figure 2.9 and Figure 2.10 flatten out as μ gets smaller for each additional component.

The results of equation 3 and 4 can be easily generalized for the hypoexponential case. First consider a hypoexponential with two stages with parameters μ_1 and μ_2 . Let x_H be

the c_H^{th} percentile of hypoexponential distribution and let x_i be the c_i^{th} percentile of the i^{th} stage, $i=1,2$. The CDF is given by:

$$(1 - e^{-\mu_1 x_H}) \frac{\mu_2}{\mu_2 - \mu_1} + (1 - e^{-\mu_2 x_H}) \frac{\mu_1}{\mu_1 - \mu_2} = c_H \quad (7)$$

Now we can find out the weight function, w , such that:

$$x_H = w(x_1 + x_2) \quad (8)$$

Again the CDF of each of the exponential component i can be written as:

$$x_i = \frac{-\ln(1 - c_i)}{\mu}$$

or

$$\mu = \frac{-\ln(1 - c_i)}{x_i} \quad (9)$$

Putting the value of x_H from equation 8 and μ_i from equation 9 in equation 7, we get

$$\begin{aligned} c_H &= \left(1 - e^{-\frac{-\ln(1-c_1)}{x_1} w(x_1+x_2)} \right) \left(\frac{\frac{-\ln(1-c_2)}{x_2}}{\frac{-\ln(1-c_2)}{x_2} - \frac{-\ln(1-c_1)}{x_1}} \right) \\ &+ \left(1 - e^{-\frac{-\ln(1-c_2)}{x_1} w(x_1+x_2)} \right) \left(\frac{\frac{-\ln(1-c_1)}{x_1}}{\frac{-\ln(1-c_1)}{x_1} - \frac{-\ln(1-c_2)}{x_2}} \right) \end{aligned} \quad (10)$$

Given x_i and c_i , $i=1,2$, we can calculate the weight function, w , using the above expression for any percentile point c_H of the two stage hypoexponential. This weight, when multiplied by the sum of x_i , $i=1,2$, gives the point x_H .

The above expression can easily be generalized as:

$$c = \sum_{i=1}^n \left\{ \left(1 - e^{-w \ln(1-c) \frac{\sum_{k=1}^n x_k}{x_i}} \right) \prod_{\substack{j=1 \\ j \neq i}}^n \frac{x_j \ln(1-c)}{x_j \ln(1-c) - x_i \ln(1-c)} \right\} \quad (11)$$

and

$$x_H = w \sum_{i=1}^n x_i \quad (12)$$

For illustration purposes we plot x_H , i.e., the c_H .th percentile of the hypoexponential distribution, calculated using its CDF formula and using the relations given by equation 10 and 11. We have set $c = 0.95$. The plots are identical for any given value of c_H . In Figure 2.11, the parameters μ_1 and μ_2 increase monotonically for each successive observation. In Figure 2.12, parameter μ_1 increases and μ_2 decreases over the same range hence giving a bowl shaped output. Both plots coincide.

Figure 2.11 95th percentile of hypoexponential-2 and equation 12

Figure 2.12 95th percentile of hypoexponential-2 and equation 12

The results hold for any number of hypoexponential stages. For illustration purposes, we have set $n=6$ and $c = 0.95$, and the parameters μ_i have been varied as follows. In Figure 2.13 $\mu_i = i * \mu_1, i = 1, \dots, 6$. In Figure 2.14, $\mu_i = \frac{i}{\mu_1}, i = 2, \dots, 6$ and in Figure 2.15, $\mu_i = i$, for $i = 1, \dots, 4$, parameters μ_5 increases and μ_6 decreases over the same range for each successive observation.

Figure 2.13 95th percentile of hypoexponential-6 and equation 12

Figure 2.14 95th percentile of hypoexponential-6 and equation 12

Figure 2.15 95th percentile of hypoexponential-6 and equation 12

2.3.3 Two-stage Coxian

The same idea can be applied to a more generalized distribution, like a two-stage Coxian distribution. We first present a brief review the Phase Type (PH) distribution.

Consider a continuous-time Markov process with $n+1$ states, where $n \geq 1$, such that the states $1, \dots, n$ are transient states and state $n+1$ is an absorbing state. Further, the process has an initial probability of starting in any of the $n+1$ phases given by the probability vector (α, α_{n+1}) , where α is a $1 \times n$ vector. Thus the PH distribution is any continuous distribution, X , on $[0, \infty)$ which can be obtained as a distribution of time until the absorption state is reached in a continuous time finite state Markov Chain. This process can be written in the form of a generator matrix as follows,

$$Q = \begin{matrix} & \begin{matrix} \text{transient} & \text{absorbing} \end{matrix} \\ \begin{matrix} \text{transient} \\ \text{absorbing} \end{matrix} & \begin{bmatrix} \mathbf{S} & \mathbf{S}_0 \\ \mathbf{0} & 0 \end{bmatrix} \end{matrix}$$

Where $\mathbf{S}$ is a $2n \times 2n$ transition rate matrix and $\mathbf{S}_0$ is defined as: $\mathbf{S}_0 = -\mathbf{S}\mathbf{1}$. $\mathbf{0}$ is a $1 \times n$ vector with each element 0 and $\mathbf{1}$ is an $n \times 1$ vector with each element 1. The pair $(\alpha, \mathbf{S})$ is called a representation function of PH distribution.

The PDF and CDF of a PH distribution are as follows:

$$f(x) = \alpha e^{\mathbf{S}x} \mathbf{S}_0$$

$$F(X) = 1 - \alpha e^{\mathbf{S}x} \mathbf{1}$$

Here $e^{\mathbf{S}x}$ represents matrix exponential, which is defined as:

$$\exp(A) = e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k = 1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots$$

A Coxian distribution, notated as C_n is a generalization of Erlang distribution and it is due to Dr. Cox. The idea behind this distribution due initially to Erlang is how one can represent an arbitrary generalized distribution in terms of a mixture of exponential distributions. Cox, considered a family of arbitrary distributions whose Laplace transform $f^*(s)$ has a rational polynomial, which he showed that it can be expressed as follows:

$$f_x^*(s) = w_0 + \sum_{i=1}^n w_i \frac{\mu_i}{\mu_i + s}$$

Where $\sum_{i=1}^n w_i = 1$ and $\mu_i, i = 1, 2, \dots, n$ are zeroes of the denominator of the rational polynomial. The distribution with the above Laplace transform can be expressed in terms of exponential stages as shown in Figure 2.16

Figure 2.16 Coxian-n distribution

One of the most commonly used Coxian distributions is the two-stage Coxian (C_2) consisting of two phases as shown in Figure 2.17

Figure 2.17 Coxian-2 distribution

The Laplace transform of C_2 is given by:

$$f_x^*(s) = a \frac{\mu_1}{\mu_1 + s} \cdot \frac{\mu_2}{\mu_2 + s} + b \frac{\mu_1}{\mu_1 + s}$$

The PDF and CDF of C_2 are given by:

$$f(x) = b\mu_1 e^{-\mu_1 x} + a \left(\frac{\mu_1 \mu_2}{\mu_2 - \mu_1} e^{-\mu_1 x} + \frac{\mu_1 \mu_2}{\mu_1 - \mu_2} e^{-\mu_2 x} \right)$$

$$F(X) = b(1 - e^{-\mu_1 x}) + a \left\{ \frac{\mu_2}{\mu_2 - \mu_1} (1 - e^{-\mu_1 x}) + \frac{\mu_1}{\mu_1 - \mu_2} (1 - e^{-\mu_2 x}) \right\}$$

Hence C_2 is a special type of PH distribution as this Markov process can be represented using the rate matrix: $S = \begin{bmatrix} -\mu_1 & a\mu_1 \\ 0 & -\mu_2 \end{bmatrix}$

Now let us consider a series of n C_2 components connected in tandem as shown in Figure 2.18

Figure 2.18 A series of Coxian 2 distributions

The end-to-end service time distribution, where each component is a C_2 , can be represented by a PH distribution with the following $2n \times 2n$ rate matrix:

$$S = \begin{bmatrix} -\mu_{11} & a_1\mu_{11} & b_1\mu_{11} & 0 & 0 & \dots & 0 & 0 \\ 0 & -\mu_{12} & \mu_{12} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & -\mu_{21} & a_2\mu_{21} & b_2\mu_{21} & \dots & 0 & 0 \\ 0 & 0 & 0 & -\mu_{22} & \mu_{22} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \dots & -\mu_{n1} & a_n\mu_{n1} \\ 0 & 0 & 0 & \dots & \dots & 0 & -\mu_{n2} \end{bmatrix},$$

and starting vector $\alpha = [1 \ 0 \ 0 \ \dots \ 0]$, of length n , the PDF and CDF of this PH distribution are:

$$f(x) = \alpha e^{Sx} S_0$$

$$F(X) = 1 - \alpha e^{Sx} \mathbf{1}$$

where $\mathbf{1}$ is a column vector of size n containing all 1's and $S_0 = -S\mathbf{1}$.

The CDF of the PH can be solved to get the value of x corresponding to a percentile c using an analytical tool such as Matlab or Mathematica.

Let x_{e2e} be the c^{th} percentile of the PH distribution corresponding to the n component C_2 distribution. Let x_i be the c^{th} percentile of the i th C_2 stage, $i=1,\dots,n$. We are looking for a weight function w such that:

$$w(x_1 + x_2 + \dots + x_n) = x_{e2e}$$

Following the same steps as before, if the triplet (μ_1, μ_2, α) is known for each component, one can calculate the weight function from the expression:

$$c = 1 - \alpha e^{Sw \sum_{i=1}^n x_i \mathbf{1}} \quad (13)$$

$$x_{e2e} = w \sum_{i=1}^n x_i \quad (14)$$

Here we assume we know the parameters of the individual C_2 components. The relation is verified in the graphs of Figure 2.19 and Figure 2.20. In Figure 2.19, we plot different percentiles calculated from original CDF and weight functions for increasing number of components where the parameters of each component are the same. Figure 2.20, we redid the experiment using different parameters for each component.

Figure 2.19 95th percentile of phase type and equation 14

Figure 2.20 95th percentile of phase type and equation 14

2.4 Inter-provider Quality of Service

A set of recommendations to simplify deployment of inter provider Quality of Service (QoS) for services spanning multiple networks were presented in a white paper [36]. It is

well known that providers currently deploy widely differentiated services based on a QoS architecture, such as DiffServ and MPLS. However, enabling QoS based peering among various providers is an area of open research and debate. The document recommends standards and best practices that can help simplify the deployment of QoS for traffic that traverses the network of various providers.

The authors considered three network performance metrics: one way latency (IPTD), one way packet loss (IPLR) and one way delay variation (IPDV) also known as jitter. Also they defined two QoS classes, a single low latency service class and a to best effort class. The low latency class is suitable for applications like VOIP and is consistent with the service class definition of Y.1541 [37]. The parameters specified in Y.1541 are as follows. IPTD is defined as the mean one way end-to-end delay with values ranging from 100 to 400 ms (depending on geographic distance). IPDV or jitter is defined as a percentile of the inter-arrival time of packets at the destinations with 50 ms being the 99th percentile, and the end-to-end packet loss is 1×10^{-3} . To support time sensitive traffic with desired QoS in a multi-provider network, the end-to-end performance metrics must be met as specified above.

The white paper lists some best practices that, if used by a substantial number of network providers, can accelerate the planning and deployment of QoS enabled networks. Most important part is to calculate maximum budgets for each network provider, such that there is a high probability that an acceptable QoS performance can be achieved across a combination of these providers providing an end-to-end service. Specifically, a fixed allocation of the end-to-end QoS metrics, referred to as the impairment budgets, to each provider is proposed. The basic idea is that the end-to-end QoS metrics are not visible to the individual provider. In view of this, a provider should not depend on a priori knowledge of other providers' networks for planning his own network and providing QoS guarantees.

Figure 2.21 7 component network model

Figure 2.21 gives a multi-domain network that was considered in the white paper. It is assumed that three concatenated core network providers would be a reasonable maximum. In addition, one access provider per end, and zero or one metro providers per end was assumed. The metro networks were not treated differently from the core networks as far as budget allocation is concerned. In view of this, the network under consideration consisted of two access networks and five core networks. This is considered to be a realistic topology for end-to-end services.

The end-to-end loss ratio and one way delay can be easily calculated by adding up the individual metrics for each provider. Each access network is allocated a loss ratio of 4×10^{-4} and each core is allocated a loss ratio of 10^{-5} , so that the end-to-end PLR is 8.5×10^{-4} which is well within the bounds specified above. Similarly for the mean delay, each access network is allocated 25 ms and each core network is allocated 10 ms, with some additional allowance for long network segments.

We are particularly interested in the budgeting of the IPDV across various segments of the network as our mathematical modeling suggest that the thresholds proposed in [36] have reasonable room for improvement. First we outline what the authors have proposed. We recall that the jitter is expressed as a percentile of the inter-arrival time at the destination.

The first observation made about IPDV is that arithmetic division will result in more stringent requirements than actually required to meet the end-to-end goals. This observation is consistent with our finding in section 2.3. Also, the authors recognized the statistical nature

of IPDV, and used probabilistic measures to allocate the metric. A major portion of the IPDV budget is allocated to the access network, where lower link speeds mandate more generous allocations. The thresholds proposed are that the 99th percentile of IPDV for core should be less than 2 ms and 99th percentile of IPDV for access should be less than 16ms. It is important to mention here that these threshold values are not calculated using a mathematical method, but they were proposed so that they can be readily achievable in core or access network. Based on these thresholds, they calculated that the end-to-end IPDV is less than 20 ms with a probability of 0.99994 in a network of five core segments, and it is less than 50 ms with a probability of 0.9998 in a network consisting of two access and five core segments. Below are the calculations presented in the paper for a five core segment network.

$$\begin{aligned}
& \text{Prob(e2e IPDV} < 20 \text{ ms)} \approx \text{Prob(Sum of IPDV thresholds} < 20 \text{ ms)} \\
& \geq \text{Prob(all 5 intervals are "Low IPDV")} + \text{Prob(4 out of 5 intervals are Low IPDV and} \\
& \text{one is High IPDV)} + \text{Prob(3 out of 5 intervals are Low IPDV and 2 are High IPDV)} \\
& = (0.99)^5 + \binom{5}{4} (0.99)^4 * (0.00999) + \binom{5}{3} (0.99)^3 * (0.00999)^2 = 0.99994
\end{aligned}$$

However, we find that the above allocations of IPDV to access and core segments are much stringent than actually required to achieve the given end-to-end bounds. As stated earlier, the IPDV thresholds or percentiles cannot be added linearly, so the statement “Prob(e2e IPDV < 20 ms) $\approx$ Prob(Sum of IPDV thresholds < 20 ms)” is not accurate mathematically.

For a five core network the document states that if 99th percentile of each core is less than 2 ms, then the end-to-end IPDV is less than 20 ms with 0.99994 probability. Let us verify the above statement considering a five component Erlang model. That is, we assume that the PDF of the one-way delay through a core network is exponentially distributed. Each of the exponential stages has a 99th percentile which is less than 2ms. Hence we can use results from section 3 to calculate the weight w such that:

$$x_{Erl} = wx_{exp}, \text{ where } x_{exp} = 2\text{ms}$$

Using equation 4:

$$\ln(1 - c_{Erl}) = w \ln(1 - c_{exp}) \ln \left(\sum_{i=0}^{n-1} \frac{(-w \ln(1 - c_{exp}))^i}{i!} \right),$$

where $c_{Erl} = 0.99994$ and $c_{exp} = 0.99$, we calculate the weight function to be:

$$w = 4.00125$$

and hence

$$x_{Erl} = 8.0025 \text{ ms}$$

That is, if we have 5 core components where the 99th percentile of each component is 2 ms, then with probability 0.99994, the end-to-end IPDV will be less than or equal to 8.0025 ms which is less than half of what was calculated in [36], i.e., 20ms.

Also, we calculate the inverse problem, i.e., given that the end-to-end IPDV is less than 20 ms with 0.99994 probability, what shall be the 99th percentile of the individual core component be? Again using equation 3 with $x_{Erl} = 20$ and $w = 4.00125$, we get:

$$x_{exp} = 5 \text{ ms}$$

That is, if we have 5 core components and we want the 99.994 percentile of the end-to-end IPDV to be 20 ms, then by setting the 99th percentile of each of the core segments to be 5 ms, we can easily guarantee the end-to-end metric (instead of 2 ms as proposed in [36]).

Next, we model the seven component network which has two access and five core networks, using a hypoexponential distribution. That is, we assume that the PDF of the jitter at each access and core network is exponentially distributed. The two access networks have the same mean, and the five core networks also have the same mean but different to that of the access networks. As proposed in the paper, the 99th percentile of IPDV for core should be less than 2 ms and the 99th percentile of IPDV for access should be less than 16ms. Using equations 11 and 12, we calculate the weight function such that:

$$x_H = w \sum_{i=1}^7 x_i$$

and hence

$$x_H = 40.8815ms$$

The end-to-end IPDV is less than equal to 40.8815 ms with probability 0.9998 (as opposed to 50 ms calculated in [36]) thus allowing 18% more of jitter for a more generous allocation at the access and core segments.

2.5 Calculating Percentiles from Actual Latency Data

As another example of the applicability of our results, we considered latency data obtained from an experiment in IBM involving signaling protocols [38]. The experiment measured the latency of two component networks and also the end-to-end latency of these two components combined together. Both components are working together to provide a Telecommunications Web Services Server (TWSS), and each component is in fact a cluster of nodes. As shown in Figure 2.22, the first component functions as an Access Gateway (AG) and the other component provides the Service Implementation (SI).

Figure 2.22 Network model

Figure 2.23, gives the histogram of the latency from the time the proxy in front of the AG cluster receives the web service (WS) request, to the time the AG node puts the WS request into the network destined for the SI cluster. Figure 2.24 gives the histogram of the latency from the time the proxy in front of the SI cluster receives the WS request, to the time the SI node puts the response into the network. And finally, Figure 2.25 gives the histogram

of the latency from the time the proxy in front of the AG cluster receives the WS request, to the time the SI puts the response into the network.

Table 2.1 Population statistics in ms

	Count	Min	Avg	Max	Std. Dev
AG latency	2750	5	29	270	22
SI latency	2750	16	42	237	15
TWSS latency	2750	32	72	312	27

Figure 2.23 AG latency data

Figure 2.24 SI latency data

Figure 2.25 TWSS latency data

We fitted hypoexponential distributions to the latency data of the individual components, i.e., AG and SI, using the moment approximation method. Hypoexponential-7 and hypoexponential-6 gave us a good approximation for AG and SI respectively.

Taking individual percentile points on the AG and SI hypoexponential approximations, we calculated end-to-end percentiles and compared them to those obtained from the actual data of the end-to-end latency. The results are given in

Table 2.2.

Table 2.2 A comparison of percentiles

Percentile	TWSS (original data)	Component Approximation
0.5	0.035081	0.037702
0.6	0.038918	0.041216
0.7	0.04334	0.046052
0.8	0.049218	0.051432
0.9	0.0589245	0.060225
0.95	0.0678105	0.067075

2.6 Single Source Shortest Path using Dijkstra's Algorithm

The expressions obtained in section III can be directly used in a search algorithm, like Dijkstra's algorithm, to calculate the shortest path in a graph that minimizes the total percentile cost (as opposed to the total average cost) of a performance metric such as, delay, energy, jitter or power attenuation of signal. Below we present an example to illustrate this.

Consider the network given in Figure 2.26. We are interested in finding the shortest path from node O to all other nodes, i.e., the minimum spanning tree (MST) rooted at node O. The link cost is the delay which we assume to be exponentially distributed.

Figure 2.26 The network under study

First we do a standard run of Dijkstra's algorithm where the cost of each link is the average delay to traverse the link and the shortest path is defined as the path with the least end-to-end 'average' delay. Figure 2.27 gives the MST (represented by dark links) rooted at node O. The average delay to reach any node from node O is given above the node.

Figure 2.27 The minimum spanning tree using the average delay

Now we use Dijkstra's algorithm to combine percentile delays. Again using Figure 2.26, we now define the per link cost to be the 95th percentile delay to traverse the link and the shortest path is defined as the path with the least end-to-end '95th percentile' delay. The addition of the percentiles is done using (11) and (12). The resultant MST is shown in Figure 2.28 (represented by dark links).

Figure 2.28 The minimum spanning tree using the percentile delay

Notice how the resultant MSTs present two very different routing views of the network. For example, consider the path from O to F. If one wants to minimize the average end-to-end delay, packets from O to F should be routed through A (Figure 2.27). This path guarantees that the average delay will be 120 time units or less. Whereas, if one is concerned about minimizing the 95th percentile delay, the packets should be routed through $A \rightarrow B \rightarrow E \rightarrow D \rightarrow G$ (Figure 2.28). This path, though having increased number of hops, guarantees that 95% of the packets will experience delay of 96.51 time unit or less.

The preference of one view over the other depends on how service level agreements (SLAs) are defined. If an SLA is defined in terms of average delays, the traditional average delay MST suffices. However, as is the case with present-day networks, for real-time communications, statistical bounding of the delay is preferred over simple averaging. In this case routing based on delay percentiles seems to be more meaningful.

2.7 Conclusions

Despite the many uses of the percentile of a random variable, very little is known as to how to add and divide percentiles. In this Chapter, we presented an exact analytic expression for adding percentiles of random variables whose PDF is a mixture of exponentials. Specifically we considered exponential, Erlang, hypoexponential, hyperexponential, and Coxian-2 distributions. We also gave two real-life examples, where we applied the results obtained here. Also we demonstrated its usefulness through the use of

an example where we employed the expressions in Dijkstra's algorithm to find the shortest path minimizing end-to-end percentile delay.

Chapter 3

3 Bandwidth Allocation for an MMPP2 Arrival Process

3.1 Introduction

Bandwidth allocation typically involves reserving part of the transmission rate of the output port of each router along the path of a connection for the traffic flow associated with the connection. The problem of allocating bandwidth under quality of service (QoS) restrictions has been analyzed extensively in the literature. Most of the proposed schemes calculate the necessary bandwidth so that the packet loss is bounded. The review of some of these techniques was given in section 1.3.

Bandwidth calculation schemes based on packet loss do not provide guarantees for the end-to-end delay. We are concerned with the calculation of the bandwidth to be allocated at the output port of each router along the path of a connection in an MPLS-enabled IP network, so that the end-to-end delay D is less than or equal to a given target delay value T

with a probability γ , i.e., $P(D \leq T) = \gamma$. In this Chapter, we assume that the arrival of packets follow an MMPP2 process. Extensions will be considered in the subsequent Chapters. We assume that the path of an LSP, i.e., an MPLS connection, has already been calculated.

In order to calculate the bandwidth so that the end-to-end delay D is less than or equal to a given value T with probability γ , we first have to calculate the end-to-end delay. A connection is typically represented by a series of queues forming a tandem queueing network, with each queue representing the output port of a router along the path. Consequently, the calculation of the end-to-end delay has been identified with the problem of calculating the end-to-end delay in a tandem queueing network. This problem has been addressed in the literature under a variety of assumptions as described in section 1.3 of Chapter 1

In this Chapter, we consider a tandem queueing network with exponentially distributed service times. The service rate at each queue represents the bandwidth μ allocated on each output port. Consequently, all service rates are the same and equal to μ . The arrival process of packets is assumed to be bursty and correlated and it is modeled by a two-stage Markov Modulated Poisson Processes (MMPP2). Using this queueing network, we calculate approximately the bandwidth μ to be allocated at each output port so that the end-to-end delay D is less than or equal to a given target delay value T with a probability γ , i.e., $P(D \leq T) = \gamma$.

The MMPP2 is a popular traffic model, because it is capable of capturing burstiness and autocorrelation characteristics commonly present in network traffic, (see e.g., Jiang and Dovrolis [39], and Figueiredo et al [40]) while satisfying a reduced complexity, compared to other processes, such as MAP. It has been used extensively in the literature to capture the correlation characteristics of multimedia sources in broadband integrated services digital networks (see e.g., Adas [41] and Kuehn [42]). Also, appropriately constructed Markov models using MMPP-2 appear to be a viable modeling tool also in the context of modeling long-range dependent traffic over several time scales as by Andersen and Nielsen in [43], Horvath and Telek in [44] and Riska and Riedel in [45] etc.

This tandem queueing network can be easily analyzed using a single-node decomposition by using the following two results: a moment-matching method proposed by Ferng and Chang [46] and a result due to Jean-Marie et al [47] who showed that the waiting time distribution in a single MMPP $_n$ /M/1 queue is n -stage hyper-exponential (H_n).

Ferng and Chang [46] used single-node decomposition to analyze a tandem queueing network with general service times where the arrival process to the first queue consisted of multiple heterogeneous MMPPs. They proposed two approximation schemes for calculating the mean end-to-end delay for a single (tagged) MMPP stream. Central to the proposed schemes is the calculation of the departure process of the tagged MMPP stream by an MMPP2. Specifically, the authors calculate the first three moments of the inter-departure time and the lag 1 autocorrelation of the successive inter-departure times, which are matched to those of an MMPP2 in order to obtain the four parameters that define an MMPP2. (more details on MMPP2 are presented in section 3.2)

The first queue is analyzed as an MMPP2/M/1 queue in isolation from the rest of the network. The departure process is characterized approximately by an MMPP2 using the moment-matching scheme, which permits us to analyze the second queue as an MMPP2/M/1 queue as well, and so on until the last queue is analyzed. The result of this decomposition is that the waiting time in each queue is characterized by a 2-stage hyper-exponential (H_2). Thus the distribution of the end-to-end delay D can be obtained by combining these H_2 distributions into a single phase-type distribution, from which we can calculate the delay percentiles and also the bandwidth that guarantees that $P(D \leq T) = \gamma$, where T and γ are given, as described in section 3.3. We implemented this single-node decomposition and compared it extensively with simulation. It turns out that this approach does not have a good accuracy because the departure process from an MMPP2/M/1 queue is not MMPP.

In view of this, we propose the following alternative method. We first construct an upper and a lower bound on a given percentile of D , from which we obtain bounds of the bandwidth such that $P(D \leq T) = \gamma$, for given T and γ . These two bounds are then combined using an interpolation function to obtain an accurate estimate of the bandwidth. The main

contribution of this Chapter is that the upper and lower bounds are constructed by analyzing only the first queue of the tandem queueing network. Thus, the proposed scheme provides results with a lower computational complexity, compared to the single node decomposition approach. Validation tests against simulation for a variety of input parameters showed that the proposed scheme is very accurate with an average relative error of 1.25%.

The Chapter is organized as follows. In section 3.2 we describe the queueing network under study, and in section 3.3 we briefly describe the single-node decomposition algorithm mentioned above. In section 3.4, we present the lower and upper bounds on the bandwidth and the interpolation function between the two bounds. Section 3.5 provides numerical results validating the proposed scheme, and the conclusions are given in section 3.6.

3.2 The queueing network under study

The queueing network under study is an open tandem network consisting of N infinite-capacity queues as shown in Figure 3.1.

Figure 3.1 The tandem queueing network under study

The arrival process is a two-state MMPP (MMPP2) characterized by the 2x2 matrices:

$$A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, Q = \begin{bmatrix} -\sigma_1 & \sigma_1 \\ \sigma_2 & -\sigma_2 \end{bmatrix}$$

where λ_i , and $1/\sigma_i$ is the rate of the Poisson arrivals and the mean value of the exponentially distributed sojourn time in state $i=1,2$, respectively. The steady state probability vector w of the Markov process associated with the infinitesimal generator Q is:

$$w = [w_1 \quad w_2] = \left[\frac{\sigma_2}{\sigma_1 + \sigma_2} \quad \frac{\sigma_1}{\sigma_1 + \sigma_2} \right]$$

The mean arrival rate is:

$$\bar{\lambda} = \lambda_1 \frac{\sigma_2}{\sigma_1 + \sigma_2} + \lambda_2 \frac{\sigma_1}{\sigma_1 + \sigma_2}$$

The main advantage of MMPP2 as a traffic model is that it can depict the burstiness and the autocorrelation of the successive inter-arrival times. The lag 1 autocorrelation function ρ is:

$$\rho = \frac{P(\Lambda - Q)^2 \Lambda (I - eP)(\Lambda - Q)^2 \Lambda e}{(2P(\Lambda - Q)^{-3} \Lambda e) - (P(\Lambda - Q)^{-2} \Lambda e)^2}$$

where

$$P = \frac{1}{w \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}} w \Lambda$$

I is the 2x2 identity matrix and e is a column vector containing ones. The lag 1 autocorrelation ρ of the MMPP2 takes values in (0, 0.5), see Casale et al [48]. The burstiness of the arrival process is measured by the squared coefficient of variation c^2 of the inter-arrival times, and is as follows:

$$c^2 = \frac{(P(\Lambda - Q)^{-2} \Lambda e) - (2P(\Lambda - Q)^{-3} \Lambda e)^2}{(2P(\Lambda - Q)^{-3} \Lambda e)^2}$$

The queueing network shown in figure 1 models the delays in an MPLS connection over an IP network. Each queue represents the queueing encountered by the packets of the connection at the output port of each router along the path of the connection. The service time at each queue is exponentially distributed with the same rate μ , which represents the

bandwidth allocated to the connection. The propagation delay between the routers is not included in the model since it is fixed.

The queueing network is used to calculate the probability density function of the one-way end-to-end delay D of the connection. Using this density function, we can obtain any percentile of the end-to-end delay distribution. The bandwidth μ for which D is less than or equal to a given target delay T with a probability γ can be found using a simple linear search as described below.

3.3 The queueing network under study

In this section we briefly summarize the single-node decomposition algorithm for analyzing the queueing network under study. As stated in section 1, the queueing network under study is analyzed using a simple nodal decomposition. Specifically, we consider the first queue in isolation as an MMPP2/M/1 queue, and we calculate the PDF of the H_2 delay in the queue (including the service time) using the results by Jean-Marie et al [47]. We also approximate its departure process by an MMPP2 using the matching-moment method by Ferng and Chang [46] to estimate parameters $\lambda_1, \lambda_2, \sigma_1$, and σ_2 (The results from [46] and [47] were applied without any modification). This departure process becomes the arrival process to the next queue, which is analyzed as an MMPP2/M/1 queue. We proceed in this fashion until all queues are analyzed. The PDF of the end-to-end delay D is obtained by combining the N individual H_2 distributions into the phase-type distribution shown in Figure 3.2. The PDF of the phase-type distribution with arrival state vector α and the transition rate matrix S , is as follows:

$$f_D(x) = \alpha e^{Sx} S_0$$

where,

$$\alpha = (1, 0, \dots, 0)$$

$$S_0 = -Se$$

$$e = [1, 1, \dots, 1]^T$$

$$S = \begin{pmatrix} -\beta_{11} & 0 & \alpha_2 \beta_{11} & (1 - \alpha_2) \beta_{11} & \dots & 0 & 0 \\ 0 & -\beta_{12} & \alpha_2 \beta_{12} & (1 - \alpha_2) \beta_{12} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -\beta_{N1} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -\beta_{N2} \end{pmatrix}$$

The corresponding cumulative density function (CDF) of D is $F_D(x) = 1 - ae^{Sx}$, from which any given percentile of the end-to-end delay can be calculated.

Figure 3.2 The phase type distribution of the end-to-end delay

The bandwidth that should be allocated at each output port, i.e., the value of μ , so that $P(D \leq T) = \gamma$ for given T and γ , is calculated using a simple linear search algorithm. We initialize the service rate at each queue μ to a random value, and then we calculate the γ percentile of D . If it is less than the target value T , then we have over-provisioned the initial bandwidth, therefore we need to reduce μ by a small step δ . If it is higher, then the initial bandwidth is under-provisioned and we need to increase μ by a small step δ . We iterate in this fashion until the absolute value of the difference between T and the γ percentile of D is less ε , where ε was set to 0.01.

Extensive comparisons of the decomposition algorithm against simulation results showed that the percentile of the end-to-end delay calculated by the decomposition algorithm, is a lower bound of the true percentile as estimated by simulation. As shown in table 1, section 5, this is due to the fact that the algorithm underestimates significantly the correlation of the departure process from each queue. As a result, the bandwidth for a

specific delay percentile is also underestimated. In view of this, we propose an alternative method based on an upper and a lower bound of the end-to-end delay.

3.4 Bandwidth estimation based on bounds

In this section, we describe a novel method for calculating the bandwidth approximately which does not require the analysis of the entire queueing network. The method is based on an upper and lower bound of the CDF of the end-to-end delay. These bounds are easy to calculate, as they are constructed by analyzing only the first queue. Using these bounds, we obtain an upper and a lower bound on the bandwidth which satisfies a given percentile of the end-to-end delay, that is, $P(D \leq T) = \gamma$ for given T and γ . The required bandwidth is then obtained by interpolating between these two bounds.

An upper bound on the CDF of the end-to-end delay D is readily provided by the CDF of the delay D_1 in the first queue. Since D is made up of the delay in the first queue as well as in the other $N-1$ queues, the probability that a customer waits more than t in the first queue is less than or equal to the probability that the end-to-end delay D of a customer is more than t , i.e., $P(D_1 > t) \leq P(D > t)$ for all t . (Stated differently, the tail of the end-to-end delay D is “higher” than the tail of D_1 .) Consequently, we have that $P(D_1 \leq t) \geq P(D \leq t)$ for all t . In addition, we have observed empirically that when the c^2 of the arrival process to the first queue is high, then $P(D_1 < t) \rightarrow P(D < t)$, as $t \rightarrow \infty$.

A lower bound can also be calculated based on the PDF of the delay D_1 of the first queue, by assuming that the arrival process to each queue i , $i=2,3,\dots,N$, of the tandem queueing network is the same as the arrival process to the first queue. Let D' be the resulting end-to-end delay, where $D' = D_1 + D_1 + \dots + D_1$. D' has a phase-type representation similar to the one in Figure 3.2 except that all the H_2 distributions are identical to that of the first queue. We have observed empirically that in this case, the tail of D' is “higher” than that of D , i.e., $P(D' > t) \leq P(D > t)$ for all t , which means that $P(D' \leq t) \geq P(D \leq t)$ for all t . This is due to the burstiness of the traffic, as measured by the c^2 of the inter-arrival time, which is reduced as the traffic goes through the queues of the tandem queueing network. This is mainly because

of the smoothing effect of the service at each queue. This drop in the c^2 values is particularly pronounced in the first queues, and specifically from queue 1 to 2, when the offered traffic to queue 1 has a high c^2 .

Figure 3.3 The CDF of D , D_1 , and D'

Let t_1 , t , and t_2 , be the delay values such that $P(D_1 \leq t_1) = \gamma$, $P(D \leq t) = \gamma$, and $P(D' \leq t_2) = \gamma$. Then, since $P(D_1 \leq t) \geq P(D \leq t) \geq P(D' \leq t)$ as $t \rightarrow \infty$, we have that $t_1 \leq t \leq t_2$. In other words, the upper (lower) bound on the CDF of D gives a lower (upper) bound on the delay for a given percentile γ . This can be seen pictorially in Figure 3.3. The CDF graphs for D , D_1 , and D' were obtained for a tandem queueing network with 10 queues. The parameters of the MMPP2 arrival process were: $\lambda_1=1$, $\lambda_2=52.7046$, $\sigma_1=0.3$ and $\sigma_2=0.4$, which result to a mean arrival rate is 23.16 and a c^2 of the inter-arrival time 20. The service rate for each node μ was set to 60. We observe that for $\gamma=0.8$, $t_1 = 0.16$, $t = 1.09$ and $t_2 = 1.32$ and therefore $t_1 \leq t \leq t_2$.

So far we have established that for fixed μ , the delay values t_1 , t and t_2 are such that $t_1 \leq t \leq t_2$. Now, let μ_1 , μ and μ_2 be the service rates such that $P(D_1 \leq T) = \gamma$, $P(D \leq T) = \gamma$, and $P(D' \leq T) = \gamma$ respectively for a given T and γ . Then, we have that $\mu_1 \leq \mu \leq \mu_2$. That is, using the upper (lower) bound on the CDF of D , we obtain a lower (upper) bound on the bandwidth. This can be easily shown as follows. Let us assume that the value of μ for which

$P(D \leq T) = \gamma$ is known, and let us call it μ_{init} . Then, using μ_{init} we calculate the PDF of the delay D_1 in the first queue. (We recall that this is an H_2 distribution which can be computed analytically using the results from Jean-Marie et al [47].) Subsequently, we calculate the delay value τ_I such that $P(D_1 \leq \tau_I) = \gamma$ and compare it to T . As discussed above, for the same percentile γ , we have that $t_1 \leq t \leq t_2$, and consequently $\tau_I \leq T$. If $T - \tau_I > 0.01$ we decrease the value of μ_{init} by a small step δ , following the simple search algorithm described in section 3, and recalculate τ_I . The decrease in the value of μ_{init} results in an increase in the delay value τ_I for the given γ . We iterate on μ_{init} until we obtain a value μ_I such that the condition $T - \tau_I < 0.01$ is satisfied. Clearly $\mu_I \leq \mu$. Following similar arguments with the upper bound, we can show that the value μ_2 for which $\tau_2 - T < 0.01$ is such that $\mu_2 \geq \mu$. Hence we have that $\mu_I \leq \mu \leq \mu_2$

An example of these two bandwidth bounds is given in Figure 3.4. The graphs were obtained for a tandem queueing network with 10 queues. The parameters of the MMPP2 arrival process are: $\lambda_1=1$, $\lambda_2=132.349$, $\sigma_1=0.3$ and $\sigma_2=0.4$, which result to a mean arrival rate is 57.3, a c^2 of the inter-arrival time 50, and a lag 1 autocorrelation of 0.376. For reference, the plots also include the bandwidth as calculated by simulation for the said network. (This was obtained using the same simple iterative procedure, but at each step the delay value t for a given γ was estimated by simulation.). As $T \rightarrow 0$, the upper and lower bandwidth bound approach infinity. This follows from the fact that as the delay requirements become more stringent, more bandwidth is required to meet them, thus leading to the limiting case of infinite bandwidth. On the other hand, as T increases, the bandwidth bounds decrease and they eventually tend to the mean arrival rate. Notice that the bandwidth requirement cannot fall below the mean arrival rate otherwise the system will become unstable. Hence the two bounds intersect again on the x axis corresponding to a very high value of T . This required bandwidth is the minimum bandwidth threshold required to keep the system stable, i.e., to keep the utilization of the first queue (and consequently of all the other queues) less than 1. We can see this in Figure 3.4 that the lower bound tends to the mean arrival rate of 57.3 as the delay T increases. (Please note that this behavior of the bounds and the simulated results holds for any percentile of the delay).

Figure 3.4 The upper and lower bounds of the bandwidth

The simulation curve showing the exact required bandwidth lies between the two bounds and it also tends to infinity as T approaches 0 and to the average arrival rate as T tends to infinity. In general, as T decreases, the simulation curve approaches the upper bound, and as T increases, the simulation approaches and eventually coincides with the lower bound. In addition, as c^2 decreases, the simulation curve becomes closer and eventually coincides with the upper bound, and as c^2 increases, the simulation curve approaches and eventually coincides with the lower bound. This behavior is consistent with the discussion above on the two bounds of the delay percentile and it can be seen in the graphs of Figure 3.5. Based on the above observations, we propose the following interpolation function $f_I(x)$:

$$f_I(x) = \mu_1(x) + (\mu_2(x) - \mu_1(x))e^{-z \log(c)}, z = \max\left(\frac{1}{N}, \log(x+1) - \frac{1}{x}\right)$$

where $\mu_1(x)$ and $\mu_2(x)$ are the bandwidth values calculated from the lower and upper bounds, N is the number of queues and c is the coefficient of variation. It should be mentioned here that this is one of the possible interpolation functions that combine the upper and the lower bounds. This particular version was selected after evaluating various possible

alternatives on the basis of resulting in least relative errors. It can be seen directly that it has three very desirable properties:

For any given number of nodes N , the factor z becomes larger as x grows, thus the combination formula tends to the single node approximation. This is desirable, as we know that asymptotically as $x \rightarrow \infty$, both bounds and the exact end-to-end delay should tend to this single node asymptotic.

For the smaller values of x (where the asymptotic regime of large x just discussed isn't in effect), z decreases as the number of nodes N increases, thus the combination formula deviates from the single node bound towards the other bound, to reflect the effect of subsequent nodes on the end-to-end delay. Again this is desirable behavior.

As the burstiness of the traffic at the first node increases (i.e., c is larger), the single node bound comes in effect from smaller values of x and/or N . Again this is desirable, because with burstier traffic most of the end-to-end delay is due to the first node. After that, the traffic is smoothed out and the subsequent $N-1$ nodes (which have the same μ as the first one) do not have a significant impact.

We compared the interpolation function with simulation results extensively, and the average error was 1.25% whereas for the decomposition algorithm was as high as 6.3%. Verification results and graphs are presented in the next section.

Figure 3.5 Bandwidth results for a tandem network with 10 queues

3.5 Validation

In this section, we compare the proposed analytic method based on the interpolation of the upper and lower bounds against simulation. We also include numerical results using the decomposition method. The simulation program was implemented in Java and it simulates the queueing network under study. The results obtained were based on a sample between 1 million and 10 million packets, depending upon the value of the squared coefficient of variation c^2 of the arrival process. The confidence intervals for the end-to-end delay percentile were computed using the batch means method. They are not shown in the figures since they are insignificant.

In Figure 3.5, we plot the bandwidth calculated using the upper and lower bounds, the interpolation method, the decomposition method, and simulation. The bandwidth was computed so that $P(\text{delay} \leq T) = 0.95$, where T varies from 1 to 30. Four different graphs are given, each for a different value of c^2 ($c^2=1, 10, 20, 30$). The tandem network under study consists of 10 queues and the arrival process to the first queue was obtained by fixing three parameters of the MMPP2 process, i.e., $\lambda_1=1$, $\sigma_1=0.3$, and $\sigma_2=0.4$, and calculating the fourth parameter λ_2 from c^2 using the c^2 formula given in section 2.

The decomposition algorithm gives good results for low values of c^2 . However, as c^2 increases, the decomposition results deviate from the simulation results and almost coincide with the lower bound. For a given c^2 the bandwidth estimated using the decomposition algorithm tends to the lower bound as T increases and eventually coincides with it. In table 1, we report the squared-coefficient of variation values for the departure process from queues 2, 5 and 10, based on the results for $c^2 = 50$ (for the arrival process). We note that the trend of the deviations of the squared-coefficient of variation of the departure process is consistent with the trend of deviations on the estimated bandwidth.

Table 3.1 Squared-coefficient of variation of the departure process.

T	Queue 2		Queue 5		Queue 10	
	Simulation	Decomp	Simulation	Decomp	Simulation	Decomp
1	50.2435	28.00154	47.9517	27.866885	45.1816	27.704413
10	29.2861	16.60698	25.0926	14.78435	21.844	14.539953
20	20.1104	11.853817	16.1835	9.854284	13.5943	9.777885
30	13.0486	9.279408	9.9354	7.506598	7.75653	7.478823

Similar results to those presented in Figure 3.5 have been obtained for tandem queueing networks of 5 and 15 queues. The trends and conclusions are similar to those in Figure 3.5. For presentation purposes we only give two graphs in Figure 3.6 and Figure 3.7, where we compare the interpolation scheme with simulation. In both cases, the parameters λ_1 , σ_1 , and σ_2 of the arrival process are the same as above and λ_2 is calculated from the

expression of the c^2 . Figure 3.6 gives results for $c^2=30$, and the bandwidth was calculated so that $P(\text{delay} \leq 20) = \gamma$, where γ is the 90th, 95th and 99th percentile. Figure 3.7 gives results for $c^2=30$, and the bandwidth was calculated so that $P(\text{delay} \leq T) = 0.95$, where $T=5,20,30$.

Figure 3.6 Interpolation function vs simulation for various delay percentiles

Figure 3.7 Interpolation function vs simulation for various values of the delay T

Figure 3.8 Interpolation function vs simulation for various c^2 values

Figure 3.9 Interpolation function vs simulation for various ρ values

Finally, in Figure 3.8 and Figure 3.9 we compare the interpolation scheme with simulation as a function of c^2 and the lag 1 autocorrelation ρ respectively. In both figures, we give the bandwidth obtained by the interpolation scheme and simulation for a queueing tandem network with 10 queues, so that $P(\text{delay} \leq T) = 0.95$, where $T = 5, 20, 30$. The parameters λ_1 , σ_1 , and σ_2 of the arrival process are the same as above and λ_2 is calculated from the expression of the c^2 in the case of Figure 3.8, and as a function of ρ (given in section 2) in the case of Figure 3.9. (Recall that the lag 1 autocorrelation of an MMPP2 takes values in $(0, 0.5)$, see Casale et al [48]) We observe that in both graphs the interpolation results track the simulation results very well.

As can be seen from the above results the interpolation method has a good accuracy. In table 2, we summarize the relative errors calculated for all the results obtained using the interpolation method and the decomposition method. The maximum relative error observed was 5.17% for the interpolation method and 13.833% for the decomposition method. The average relative error observed was 1.25% for the interpolation method and 6.29% for the decomposition method

Table 3.2 Relative Errors

Relative Error	Interpolation	Decomposition
Maximum	5.1676%	13.833%
Minimum	0%	0%
Average	1.252%	6.287%

We conclude this section, by presenting an example where the arrival process is an Interrupted Poisson Process (IPP). This process is obtained from the MMPP-2 by setting the arrival rate in one of the two stages to zero. As a result, the process consists of alternating ON and OFF states. Let λ be the rate of arrivals during the ON state, and let $1/\alpha$ and $1/\beta$ be the mean value of the exponentially distributed duration of the ON and OFF state respectively. The IPP can be used to model the traffic stream generated by a Voice over IP (VoIP) call. VoIP calls follow a similar ON and OFF representation but with deterministic arrivals. That is, the packet inter-arrival time during the ON period is constant, equal to the packetization delay of the voice codec. Simulation results (not reported here) have shown that assuming exponentially distributed inter-arrival times during the ON period, as is the case of the IPP, with a mean equal to the packetization delay, gives a very tight upper bound on the percentile delay obtained assuming constant inter-arrival times during the ON period. Resultantly, the bandwidth calculated to satisfy a delay percentile with an IPP arrival process also provides a tight upper bound on the bandwidth required by a VoIP stream to satisfy a similar end-to-end percentile delay.

The parameter values chosen for the IPP stream are set equal to the parameter values of a VoIP call. A VoIP has, on average, 400 ms of ON period, 600 ms of OFF period [6], and the number of packets generated per second during the ON period is 50. A voice payload of 160 bytes is carried in a separate IP packet with a 40-byte IP/UDP/RTP header. Thus, we have $\alpha=2.5$, $\beta=1.667$ and $\lambda=50$. We consider a similar network as before, consisting of 10 nodes, and calculate the bandwidth required to satisfy the 95th percentile delay, where the 95th percentile delay varies from 20 ms to 200 ms. This particular range of delays was

selected keeping in mind the recommended bounds on the one-way delay for a voice conversation [6]

The results obtained for a 10-node tandem queueing network are shown in Figure 3.10. Again the interpolation gives good accuracy with a maximum relative error 5.2% and an average relative error 2.16%. Let us now consider a VoIP call between two end-points with a 30 ms propagation delay. In order for the 95th percentile of the total end-to-end delay to be 150 ms, the 95th percentile of the sum of the queueing delays has to be 130 ms. From Figure 3.10, we see that this can be achieved by allocating a bandwidth to each queue equal to 254Kbps.

Figure 3.10 Bandwidth results for a tandem network with 10 queues for a single IPP.

3.6 Conclusions

In this Chapter, we propose a simple and efficient scheme for estimating the bandwidth that should be allocated on each link of an MPLS connection in an IP network so that the end-to-end delay D is bounded statistically. That is, D is less than or equal to a given

value T with a probability γ . The arrival process is assumed to be bursty and correlated and it is depicted by an MMPP2, and the service times are exponentially distributed. Of interest is that this scheme requires the calculation of an upper and lower bound of the bandwidth by analyzing only the first queue. Extensive comparisons with simulation shows that the proposed method has an average relative error of 1.25%. We also implemented a decomposition algorithm using known results from which the bandwidth can also be estimated. This decomposition algorithm is a fairly common method for analyzing queueing networks that do not have product-form solution. We show that this decomposition does not have a good accuracy in addition to being more CPU intensive than our proposed scheme.

Chapter 4

4 Bandwidth Allocation for a MAP2 Arrival Process

4.1 Introduction

Open queueing networks are a popular modeling tool for the performance analysis of computer and telecommunication systems. Exact solution methods are available only for networks with Poisson traffic input, specific service time distribution and service discipline. These restrictive assumptions make the exact solutions unlikely to use in practice. The main reason is that in real systems the Poisson process is usually not a good model for the traffic behavior. Instead the real traffic can be bursty and correlated, and the service times in the service stations can be correlated as well. Since these features have an impact on the performance measures, they have to be taken into consideration.

Several modeling approaches were developed to better describe the properties of packet traffic. One of the lines of research is based on Markovian models with the aim of

extending the Poisson arrival process in order to capture more statistical properties of the traffic behavior. A long series of efforts resulted in the introduction of Markov arrival processes (MAPs). The MAP was first named by Lucantoni et al. [49]. But, the MAP had already been studied under different names. The versatile Markovian point process of Neuts and the N-process of Ramaswami were found to be equivalent to the BMAP (Batch Markovian Arrival Process) which is an extended version of MAP. (see [50] by Lucantoni for an introduction)

A main advantage of using Markovian models for traffic description of queues is that there are efficient numerical analysis methods, commonly referred to as matrix analytic methods, for the evaluation of a Markovian queue (see e.g., [<http://www.win.ua.ac.be/~vanhoudt/>] for MATLAB toolbox for solving quasi-birth-and-death, M/G/1, GI/M/1 and non-skip-free type markov chains by B. V. Houdt.). Also the MAP can represent a variety of processes which includes, as special cases, the Poisson process, the phase-type renewal processes, the MMPP (Markov Modulated Poisson Process) and superposition of these.

The end-to-end delay is the time it takes to deliver a packet from the transmitter to the receiver. It is made up of a fixed component and a variable component. The fixed component is the sum of all fixed delays that a packet encounters from the transmitting end-device to the receiving end-device, such as, propagation delay, fixed delays induced by transmission systems, and fixed switch processing times. The variable component is the sum of all variable delays that a packet encounters from the transmitting device to the receiving device. These delays are primarily due to queueing delays in the routers along the packet's path. The end-to-end delay is often measured in terms of its mean, but it is more meaningful to express it as a statistical upper bound in the form of a percentile. That is, we say that the end-to-end delay is less than, say 150 ms, 95% of the time. The mean is a popular measure of the end-to-end delay but it does not capture its variability, and therefore, it is not a very useful metric. On the other hand, a delay percentile gives a better idea of the variability of the end-to-end

delay and provides a bound which is valid a given percent of the time, such as, 95% and 99%.

In this chapter, we are concerned with the calculation of a percentile of the end-to-end delay of an MPLS connection in an IP network. A connection is typically represented by a series of queues forming a tandem queueing network, with each queue representing the output port of a router along the path. Consequently, the calculation of the end-to-end delay is identified with the problem of calculating the end-to-end delay in a tandem queueing network.

This problem has been addressed in the literature under a variety of assumptions. A common approach is to calculate a bound on the end-to-end delay, see Charara and Scharbarg et al [25], Koj et al [26], and Goyal et al [29]. An exact numerical expression of the end-to-end delay in a tandem queueing network can also be obtained by calculating its Laplace transform and then inverting it numerically to obtain delay percentiles. This approach was used by Xiong and Perros [32] within the context of resource optimization of web services using a tandem queueing network with Poisson arrivals and exponentially distributed service times. Also, Vleeschauwer et al [27] obtained the moment generating function of the end-to-end delay of N statistically independent M/G/1 nodes which was then inverted numerically in order to compute delay percentiles. Yeung and Lehoczký [30] used a Brownian process to study a two-stage queueing network with customers having deadlines (constant and uniformly distributed) and calculated bounds for two different scheduling disciplines, early-deadline first and FIFO. Fractional Brownian Motion (FBM) was used by Lelarge et al [34] to show that the end-to-end delay of a tagged flow in a tandem queueing network, and more generally in a tree network, is completely dominated by the queue with the maximal Hurst parameter. Feng and Chang [46] used single-node decomposition to analyze a tandem queueing network with general service times where the arrival process to the first queue consisted of multiple heterogeneous MMPPs (Markov Modulated Poisson Processes) in order to obtain the mean end-to-end delay. Central to the single-node decomposition was the calculation of the departure process from each queue by an MMPP2.

This was done by matching the first three moments of the inter-departure time and the lag 1 autocorrelation of the successive inter-departure times to those of an MMPP2 in order to obtain the four parameters that define an MMPP2. As shown in Chapter 3, this single-node decomposition method does not work well, because the squared coefficient of variation of the inter-departure time of each queue analyzed in isolation is considerably underestimated. In view of this, the authors proposed the following alternative method to estimating a given percentile of the end-to-end delay T . An upper and a lower bound on a given percentile of T were constructed by analyzing only the first queue of the tandem queueing network. These two bounds were then combined using an interpolation function to obtain an accurate estimate.

In this chapter, we extend the approach described in Chapter 3 to the case where the arrival process is a two-state Markov arrival processes (MAP), see Lucantoni et al. [50][49]. A MAP can represent a variety of processes that includes, as special cases, the Poisson process, the phase-type renewal processes, the MMPP and superposition of these. As will be seen below, we also used a MAP2 to successfully approximate the packet arrival process of various video streams, such as Cisco's Telepresence, IPTV and WebEx.

The chapter is organized as follows. In the next section we describe the tandem queueing network under study and review some of the basic properties of MAPs. In section 3 we first describe the lower and upper bounds of a given percentile of the end-to-end delay T , and then we give the interpolation function between the two bounds. Section 4 provides numerical results validating the proposed scheme. In section 5, we estimate the delay percentile of three traces, namely, Telepresence, IPTV, and WebEx, and we calculate the bandwidth required to be allocated on each link along an MPLS connection so that the 95th percentile of the end-to-end delay is bounded by a given delay. Finally, the conclusions are given in section 6.

4.2 The queueing network under study

The queueing network under study is an open tandem network consisting of N infinite-capacity queues as shown in Figure 4.1. The queueing network shown below models the delays in an MPLS connection over an IP network. Each queue represents the queueing encountered by the packets of the connection at the output port of each router along the path of the connection. The service time at each queue is exponentially distributed with the same rate μ , which represents the bandwidth allocated to the connection. The propagation delay between the routers is not included in the model since it is fixed.

Figure 4.1 The tandem queueing network under study

The arrival process is a two-state MAP (MAP2). A MAP is a process that counts transitions of a finite continuous-time Markov chain with m states. The size m is called the order of the MAP, and determines the dimensions of matrices D_0 and D_1 :

$$D_0 = \begin{pmatrix} -q_{11} & q_{12} & \cdots & q_{1m} \\ q_{21} & -q_{22} & \cdots & q_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \cdots & -q_{mm} \end{pmatrix}, D_1 = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{pmatrix}$$

where $q_{ii} = \sum_{j=1, j \neq i}^m q_{ij} + \sum_{j=1}^m a_{ij}$. D_0 and D_1 represent the transition rates of the MAP process and define the infinitesimal generator $D = D_0 + D_1$. D_0 has negative diagonal elements and non-negative off-diagonal elements governing the transitions of the arrival process that do not produce an arrival and D_1 is a non-negative matrix, with elements governing the transitions of the arrival process which produce an arrival. The MAP2 process is uniquely defined by six parameters because the negative diagonal elements of D_0 are

expressed using the remaining parameters: $q_{11} = q_{12} + a_{11} + a_{12}$ and $q_{22} = q_{21} + a_{21} + a_{22}$. The steady state probability vector π for a MAP2 process is defined as follows [51][52]:

$$\pi(1,1) = \frac{a_{21} + q_{21}}{a_{21} + q_{21} + a_{12} + q_{12}}, \pi(1,2) = \frac{a_{12} + q_{12}}{a_{21} + q_{21} + a_{12} + q_{12}}$$

The six parameter characterization of a MAP2 process permits finer control for capturing the properties of the original process that heavily influence the performance of a queueing system, as opposed to four parameter characterization (MMPP2) or one (Poisson) and, thus, it can achieve a much closer match to the original process.

The average rate of arrival (the number of arrivals in a unit time) in a MAP is called the fundamental rate of the MAP. The fundamental rate of MAP (D_0, D_1) is given by

$$\lambda = \pi D_1 e$$

where π is the stationary probability vector in the Markov chain with infinitesimal generator $D = D_0 + D_1$ (i.e., $\pi D = 0$ and $\pi e = 1$)

The marginal distribution of the inter-arrival time of the above MAP is a $\text{PH}(\alpha, D_0)$ distribution where, $\alpha = \pi D_1 / \lambda$, which is the stationary probability vector immediately after the arrival. Since the marginal distribution of the interarrival time in a MAP is a PH distribution, its moments, density and distribution function can be calculated using the standard formulae of a Phase Type distribution.

For the interval stationary MAP, we have the following inter-arrival time distribution function $F(x) = \Pr\{X \leq x\}$:

$$F(x) = \alpha(I - e^{D_0 x})(-D_0)^{-1} D_1 e,$$

and the following moments of the inter-arrival time X :

$$m_k \equiv E\{X^k\} = k! \alpha(-D_0)^{-(k+1)} D_1 e, k = 1, 2, \dots$$

From the joint distribution function, we have the autocovariance function $\psi[k]$ ($k \geq 1$) of the inter-arrival times

$$\psi[k] \equiv E[\{X_i - m_1\}\{X_{i+k} - m_1\}] = \alpha D_0^{-2} D_1 [T^{k-1} - ep] D_0^{-2} D_1 e$$

From [10] and [11] we have the Laplace transform of the virtual waiting time $w(s)$ of a MAP2/G/1 queue, which can be simplified to the following expressions for the MAP2/M/1 case:

$$w(s) = s(1 - \rho)g[sI + D_0 + D_1 H(s)]^{-1}$$

where $H(s)$ is the Laplace transform of the service time, i.e.

$$H(s) = \frac{\mu}{\mu + s}$$

$$D_0 = \begin{bmatrix} -(a + b + c) & a \\ d & -(d + e + f) \end{bmatrix}, \quad D_1 = \begin{bmatrix} b & c \\ e & f \end{bmatrix}$$

$$g = \begin{bmatrix} \frac{G_1}{G_0 + G_1} & \frac{G_0}{G_0 + G_1} \end{bmatrix}$$

$$G_0 = 1 - G_1 - H(a + bG_0 + c(1 - G_1) + d + e(1 - G_0) + fG_1)$$

$$G_1 = \frac{dG_0 + e(1 - G_0)G_0 + cG_1^2}{a + c + (b - f)G_0}$$

and e is a column vector of 1's.

4.3 End-to-end delay estimation based on bounds

In this section, we describe a method for calculating the end-to-end percentile delay approximately which does not require the analysis of the entire queueing network. The method is based on an upper and lower bound of the cdf of the end-to-end delay. These bounds are easy to calculate, as they are constructed by analyzing only the first queue. Using

these bounds, the required end-to-end percentile delay is obtained by interpolating between these two bounds.

An upper bound on the cdf of the end-to-end delay T is readily provided by the cdf of the delay T_1 in the first queue. Since T is made up of the delay in the first queue as well as in the other $N-1$ queues, the probability that a customer waits more than t in the first queue is less than or equal to the probability that the end-to-end delay T of a customer is more than t , i.e., $P(T_1 > t) \leq P(T > t)$ for all t . (Stated differently, the tail of the end-to-end delay T is “higher” than the tail of T_1 .) Consequently, we have that $P(T_1 \leq t) \geq P(T \leq t)$ for all t . In addition, we have observed empirically that when the c^2 of the inter-arrival times of the arrival process to the first queue is high, then $P(T_1 < t) \rightarrow P(T < t)$, as $t \rightarrow \infty$. To construct the upper bound on the cdf of the end-to-end delay, we require the pdf of the delay T_1 in the first node, which is a MAP2/M/1 queue. The pdf values were obtained using the Laplace transform $w(s)$ given above.

A lower bound can also be calculated based on the pdf of the delay T_1 of the first queue, by assuming that the arrival process to each queue i , $i=2,3,\dots,N$, of the tandem queueing network is the same as the arrival process to the first queue. Let T' be the resulting end-to-end delay, where T' is the of N -fold convolution of the delay T_1 . We have observed empirically that in this case, the tail of T' is “higher” than that of T , i.e., $P(T' > t) \leq P(T > t)$ for all t , which means that $P(T' \leq t) \geq P(T \leq t)$ for all t . This is due to the burstiness of the traffic, as measured by the c^2 of the inter-arrival time, which is reduced as the traffic goes through the queues of the tandem queueing network. This is mainly because of the smoothing effect of the service at each queue. This drop in the c^2 values is particularly pronounced in the first queues, and specifically from queue 1 to 2, when the offered traffic to queue 1 has a high c^2 .

To construct this lower bound we require the N -fold convolution of the delay T_1 of the first node. This leads to complex expressions if done using the Laplace transform $w(s)$ given above. Instead, we approximate the waiting time distribution in a MAP2/M/1 queue

with a Gamma distribution. The N -fold convolution of delay T_1 is then easily obtained using the summation property of the Gamma distribution.

The Gamma distribution is a two-parameter family of continuous probability distributions. It has a shape parameter α and a scale parameter $1/\beta$. If α is an integer, then the distribution represents an Erlang distribution, i.e., the sum of α independent exponentially distributed random variables, each of which has a rate of β . The pdf of a Gamma-distributed random variable x is given as:

$$f(x; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, x \geq 0$$

From the Laplace transform $w(s)$ given above, we obtain the first two moments m_1 and m_2 of the waiting time T_1 in the first queue. We then construct an equivalent Gamma distribution with α and β calculated as follows:

$$\alpha = \frac{m_1^2}{m_2 - m_1^2}, \beta = \frac{m_2 - m_1^2}{m_1}$$

The N -fold convolution of a Gamma distribution is obtained using the summation property. That is, let X_i be a Gamma distributed random variable with parameters α and $1/\beta$, $i=1,2,\dots,N$ (i.e., all random variables have the same scale parameter). Then

$$\sum_{i=1}^N X_i \sim \text{Gamma} \left(\sum_{i=1}^N \alpha_i, 1/\beta \right)$$

4.3.1 The Interpolation Function

An example of the upper and lower bounds of the 95th percentile of the end-to-end delay T is given in Figure 4.2. The graphs were obtained for a tandem queueing network with 10 queues. The parameters of the MAP2 arrival process are: (0.1, 0, 0.1, 1, 5, 54), which result to a mean arrival rate 2, a c^2 of the inter-arrival time 18.56, and a lag 1 autocorrelation of -0.02. As the utilization U tend to 1, the upper and lower bounds approach infinity. This

follows from the fact that as the load increases, the delay incurred becomes larger and consequently the 95th percentile also gets larger, thus leading to the limiting case of infinity. On the other hand, as U decreases, the delay bounds decrease and the 95th percentile of T tends to 0. Hence the two bounds intersect again on the x axis corresponding to a very low value of U . (Please note that this behavior of the bounds and the simulated results holds for any percentile of the delay T).

Figure 4.2 The upper and lower delay bounds

The simulation curve showing the exact 95th percentile delay lies between the two bounds and it also tends to infinity as U approaches 1 and to 0 as U tends to 0. In general, as U increases, we have observed empirically that the simulation curve approaches the lower bound, and as U decreases, the simulation approaches and eventually coincides with the upper bound. This is because when the network utilization is low, the end-to-end delay is the sum of the delays incurred in all nodes. However, as network utilization increases, most of the delay T is due to the delay incurred in the first node, with little or no delay in the following nodes. This is because the exponential server of the first node acts as a smoothing function on the arrival process. The first node absorbs most of the burstiness and resultantly incurs the maximum delay. In addition, we have observed empirically that as c^2 decreases, the simulation curve becomes closer and eventually coincides with the upper bound, and as c^2 increases, the simulation curve approaches and eventually coincides with the lower bound.

This is because of reason similar to those identified above. Based on the above observations, we propose the following interpolation function $f_I(x)$:

$$f_I(x) = \eta_1(x) + (\eta_2(x) - \eta_1(x))e^{-z},$$

where

$$z = \log(c) - \log(-\log(\max[1 - U, \rho])),$$

$\eta_1(x)$ and $\eta_2(x)$ are the delay values calculated from the lower and upper bounds, U is the utilization of the first node, c is the coefficient of variation and ρ is the autocorrelation. It should be mentioned here that this is one of the possible interpolation functions that combine the upper and the lower bounds. This particular version was selected after evaluating various possible alternatives on the basis of resulting in the least relative error. It can be seen directly that it has the following very desirable properties:

The factor z becomes larger as U grows, thus the combination formula tends to the single node approximation. This is desirable, as we know that asymptotically as $U \rightarrow 1$, both bounds and the exact end-to-end delay tend to this single node asymptotic.

As the burstiness of the traffic at the first node increases (i.e., c gets larger), the single node bound comes in effect even for smaller values of U . Again this is desirable, because with burstier traffic most of the end-to-end delay is due to the first node. After that, the traffic is smoothed out and the subsequent $N-1$ nodes (which have the same μ as the first one) do not have a significant impact.

We compared the interpolation function with simulation results extensively, and the average error was 4.24%. Verification results and graphs are presented in the next section.

4.4 Validation

In this section, we compare the proposed approximation method against the exact solution obtained by simulation. The simulation program was implemented in Java and it

simulates the queueing network under study. The results obtained were based on a sample between 1 million and 10 million packets, depending upon the value of the squared coefficient of variation c^2 of the inter-arrival times of the arrival process. The confidence intervals for the end-to-end delay percentile were computed using the batch means method. They are not given in the figures since they are extremely small.

The results are presented in a series of graphs given in Figure 4.3. For each graph we plotted the upper and lower bound, the exact solution obtained by simulation, and the approximation results obtained by interpolating between the upper and lower bounds of the 95th percentile of the end-to-end delay T for different utilization values varying from 0.1 to 0.9. All the results are for a 10 node tandem queueing network. Each figure corresponds to a MAP2 with different autocorrelation lag-1 ρ and c^2 . The average arrival rate λ is obtained from the parameters of a MAP2 using the expressions in section 2. The utilization levels were obtained by varying the value of μ .

The values for the lag-1 autocorrelation ρ and c^2 were obtained from the set of all feasible values for ρ and c^2 for a MAP2. The lag-1 autocorrelation ρ for a MAP2 varies from -0.5 to 0.5 [52]. For negative values of ρ we found experimentally that c^2 can only take low values. This makes sense intuitively because as inter-arrival times become inversely related to each other (smaller ones followed by larger ones and vice versa), traffic bursts reduce. For positive values of ρ , c^2 can take any (positive) value. In addition, for high values of positive autocorrelation, c^2 cannot attain very low values. This makes intuitive sense because as the inter-arrival times become inter-dependent, the burstiness of the traffic increases as well.

Below we give graphs for a representative set of values of ρ and c^2 .

Figure 4.3 95th Percentile Delay results for a Tandem Network with 10 queues

In the first four graphs the value of c^2 is very low and the simulation and approximation curves closely follow the upper bound. That is, the end-to-end delay closely follows the sum of the delays of the individual nodes. Having a low c^2 or a negative autocorrelation both have similar effects on the arrival process i.e., reduced burstiness. Similarly, for high c^2 and/or ρ values, the simulation and approximation curves match more closely to the lower bound. This is because as the burstiness of the traffic increases, most of the delay incurred is due to the first node and the delay components of the remaining nodes fade in comparison to the first node.

As can be seen from the above results the interpolation method has a good accuracy. The maximum relative error observed was 8.56%, the minimum relative error observed was 0% and the average relative error observed was 4.24%.

4.5 Video Traces

In this section, we use the approximation method described above to estimate the 95th percentile of the end-to-end delay T of video traces and also calculate the bandwidth required such that the 95th percentile of the T is less than a specified value T^D . Three different types of traces were used, namely, Cisco point-to-point Telepresence, IPTV and WebEx [53]. The packet arrival process of the trace was characterized by a MAP2 using the results given by Heindl et al in [52]. Briefly, the inter-arrival time distribution of a MAP2 is a phase type distribution whose distribution function can be written as a matrix-exponential as follows:

$$F(x) = 1 - P_{ME} e^{-B_{ME}x} e_{ME}, \text{ for } x \geq 0$$

A correlated matrix exponential sequence can be constructed from moment and correlation data. In two dimensions, which is the dimension for a MAP2, the resulting moment/correlation-canonical form is given by:

$$P_{ME} = [1 \quad 0], \quad B_{ME}^{-1} = \begin{bmatrix} r_1 & r_1 \\ \frac{r_2 - r_1^2}{r_1} & \frac{r_3 - r_1(2r_2 - r_1^2)}{r_2 - r_1^2} \end{bmatrix}$$

$$e_{ME} = [1 \quad 0]^T, Y_{ME} = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix}$$

Where $r_i = \frac{E[X^i]}{i!}$, $i = 1, 2, 3$ are the first three reduced moments of the marginal distribution. Also, in matrix Y_{ME} , the parameter ρ is used to characterize the autocorrelation structure.

The Cisco Telepresence trace has a lag-1 autocorrelation $\rho = -0.0729$, $c^2 = 8.1878$ and an arrival rate $\lambda = 7.13$ Mbps. The distribution of the packet sizes and the arrival rate (Mbps) are given in Figure 4.4. We observe that the packet size varied from 60 B to 1260 B, and the arrival bit rate is almost constant.

Figure 4.4 Telepresence Trace. Packet Length Distribution (left), Arrival Bit Rate (right)

Following the approximation method described in section 3, we first constructed the upper and lower bounds of the 95th percentile of the end-to-end delay, for various utilization levels ranging from 0.1 to 0.9 obtained by varying μ from 71.3 Mbps to 7.92 Mbps. A 10-node network was assumed. The 95th percentile of the end-to-end delay was subsequently obtained by interpolation as described in section 3.1. In Figure 4.5(left), we give the upper and lower bounds, the approximate results, and also the exact results obtained by simulation as a function of the server utilization which represents the link utilization in a network. The maximum relative error observed was 8.2%, the minimum relative error observed was 0% and the average relative error observed was 4.5%.

The minimum bandwidth required so that the 95th percentile of the end-to-end delay T is equal to a given value T^D , can be obtained using the following simple search algorithm. We initialize the service rate at each queue μ to a random value, and then we calculate the 95th percentile of T . If it is less than the target value T^D , then we have over-provisioned the initial bandwidth, and therefore we need to reduce μ by a small step δ . If it is higher, then the initial bandwidth is under-provisioned and we need to increase μ by a small step δ . We iterate in this fashion until the absolute value of the difference between T^D and the 95th percentile of T is less than ε , where ε was set to 0.01. The required bandwidth as a function of T^D is given in the right plot of Figure 4.5. The curve labeled “exact” gives the exact result as calculated by simulation and the curve labeled “approximation” gives the results obtained with the approximation method. We can see that for very large delay percentiles the bandwidth requirement decreases, but in the limiting case we need at least as much bandwidth as the average arrival rate. The bandwidth requirement cannot fall below the average arrival rate as that will deem the queue unstable. The relative error between the exact and approximation results was observed to be less than 5%.

Figure 4.5 Telepresence Trace. The 95th Percentile of the T vs. Utilization (left), Bandwidth Required vs. T^D (right)

The IPTV trace has a lag-1 autocorrelation $\rho = -0.25187$, $c^2 = 3.6558$ and an arrival rate $\lambda = 13.03$ Mbps. The distribution of the packet sizes and the arrival rate (Mbps) are given in

Figure 4.6. We observe that the trace had a small number of signaling packets, and the bulk of the traffic was video packets of size 1324B.

Figure 4.6 IPTV Trace. Packet Length Distribution (left), Arrival Bit Rate (right)

Again, following the approximation method described in section 3, we first constructed the upper and lower bounds of the 95th percentile of the end-to-end delay, for various utilization levels ranging from 0.1 to 0.9 obtained by varying μ from 130.3 Mbps to 14.78 Mbps. A 10-node network was assumed. The 95th percentile of the end-to-end delay was subsequently obtained by interpolation as described in section 3.1. In Figure 4.7(left), we give the upper and lower bounds, the approximate results, and also the exact results obtained by simulation as a function of the server utilization which represents the link utilization in a network. The maximum relative error observed was 3.22%, the minimum relative error observed was 0% and the average relative error observed was 1.35%.

The required bandwidth as a function of T^D is given in the right plot of Figure 4.7. The relative error between the exact and approximation results was observed to be less than 5%.

Figure 4.7 IPTV Trace. The 95th Percentile of the T vs. Utilization (left), Bandwidth Required vs. T^D (right)

The WebEX trace has a lag-1 autocorrelation $\rho = 0.0486$, $c^2 = 51.0135$ and an arrival rate $\lambda = 0.309$ Mbps. The distribution of the packet sizes and the arrival rate (Mbps) are given in Figure 4.8

Figure 4.8 WebEx Trace. Packet Length Distribution (left), Arrival Bit Rate (right)

In Figure 4.9 (left), we give the upper and lower bounds, the approximate results, and also the exact results obtained by simulation as a function of the server utilization which represents the link utilization in a network. The maximum relative error observed was 6.62%, the minimum relative error observed was 0% and the average relative error observed was 3.15%. The required bandwidth as a function of T^D is given in the right plot of Figure 4.9. The relative error between the exact and approximation results was observed to be less than 5%.

Figure 4.9 WebEx Trace. The 95th Percentile of the T vs. Utilization (left), Bandwidth Required vs. T^D (right)

4.6 Conclusions

In this chapter we proposed an efficient and accurate approximation method for calculating a given percentile of the end-to-end delay along a path of a point-to-point MPLS connection. That is, we calculate a value T^D such that the end-to-end delay T is less than or equal to T^D for a given probability γ , i.e., $P(T \leq T^D) = \gamma$.

The MPLS connection is modeled by a tandem queueing network of infinite capacity queues. The arrival process of packets to the connection is assumed to be bursty and correlated and it is depicted by a two-stage Markovian Arrival Process (MAP2). The service times are exponentially distributed. The proposed method uses only the first queue of the tandem queueing network to construct an upper and lower bound of a given percentile of the end-to-end delay, which we approximate using an interpolation function between the two bounds. Extensive comparisons with simulation showed that the results obtained have a maximum relative error of 8.56% and the average relative error observed was 4.24%.

We used this method to estimate the 95th percentile of the end-to-end delay of various video traces, such as Cisco's point-to-point Telepresence, IPTV, and WebEx. Then using a simple linear search, we obtained the minimum amount of bandwidth required to be allocated on each link along the path of a point-to-point MPLS connection carrying one of

the video traces, so that a given 95th percentile of the end-to-end delay is satisfied. The relative errors were observed to be less than 5%.

Chapter 5

5 Bandwidth Allocation of a Video Stream using Traces

5.1 Introduction

Video traffic is widely expected to account for a large portion of the traffic in future wired and wireless networks. For interactive video the end-to-end delay has to be less than 200 msec in order to guarantee good QoE. Interestingly enough, little is known about how much bandwidth should be allocated to an interactive video so that a given percentile of the end-to-end delay is satisfied. In chapter 4, we presented an approximation algorithm for calculating a given percentile of the end-to-end delay of a video connection which is characterized by a packet trace. The end-to-end delay is depicted by a tandem queueing network of infinite capacity queues. The video trace is approximated by a two-stage Markovian Arrival Process (MAP2), which is the arrival process to the tandem network. The proposed method uses only the first queue of the tandem queueing network to construct an

upper and lower bound of a given percentile of the end-to-end delay. The percentile value is then approximated by interpolating between the two bounds. Using this method together with a simple search procedure, we obtained the minimum amount of bandwidth required to be allocated on each link along the path of the video stream, so that a given percentile of the end-to-end delay is satisfied. However, the algorithm described above for calculating a percentile of the end-to-end delay in a tandem queueing network, cannot handle the presence of background traffic that competes with a tagged video within the output port of a router. In view of this, in this chapter we propose an exact and efficient algorithm for calculating any percentile of the end-to-end delay in the presence of background video traffic, assuming that all video flows are characterized by packet traces and an arriving packet keeps its packet length in the entire queueing network. These traces may be a single stream or multiple streams. Based on this algorithm, the minimum amount of bandwidth required, so that a given percentile of the end-to-end delay is satisfied, is easily obtained using a simple search algorithm.

5.2 Literature review

The algorithm proposed in this Chapter deals with the problem of studying the queueing behavior of an arrival stream, referred to as the *tagged process*, in the presence of other streams, and referred to as the *background stream*. This is a difficult topic that has not been addressed adequately within the context of a tandem queueing network.

The inter-departure time of the tagged stream from a single queue in the presence of background traffic has been studied in the literature. For instance, the exact Laplace transform of the class-dependent inter-departure time distribution in a multi-class queue, where each arrival process is Poisson and the service time has a class-dependent general distribution was obtained by Stanford and Fischer [54]. Dasu [55] obtained a closed-form expression of the Laplace transform of the inter-departure time of the tagged traffic in a two-class single server queueing system where the tagged arrival process is a generalized phase process, the background arrival process is Poisson, and the service time follows a phase-type

distribution. Several approximations have also been reported under a variety of assumptions. Whitt [56] developed two moment approximations of the departure process of a single class of customers in a multi-class GI/G/m queue. In Kumaran et al. [57], the tagged and the background arrival processes were assumed to be matrix exponential (ME), and the service time distribution was also an ME. The authors obtained an approximation for the tagged departure process. In Mitchell et al. [58], an approximation of the tagged departure process was also obtained for heavy and light traffic under similar assumptions as the previous paper. The above references are for continuous-time models. In addition, the problem of determining the inter-departure time has been also considered in the discrete-time domain for ATM networks, see for instance Sohraby and Privalov [59]. More recently, Geleji [60] obtained an exact numerical solution of the inter-departure time of a tagged process from a single queue, assuming that both the tagged arrival process and the background arrival process are MMPP, and that the service time is exponentially distributed. Under similar assumptions, Geleji also gave an analytic expression that is an upper bound of the tagged inter-departure time from a tandem queueing network of any number of nodes.

Wang et al. [61] studied the per stream short term loss behavior of multimedia traffic. They modeled both the tagged multimedia and background traffic by a discrete-time Markovian arrival process and discrete-time batch Markovian arrival process respectively and modeled the router as a DMAP+DBMAP/D/1/K queueing model. However, their work focuses solely on the short-term characteristics of packet drop and not the end to end delay. In [62], Stavrakakis formulated a queueing system to study the distortion induced in a tagged ATM stream. A discrete-time analysis in the M/G/1 paradigm yielded the numerical results for cell delay, delay jitter and inter-departure time probability distributions at an ATM multiplexer. Similarly, Conti et al. [63] evaluated the impact of temporal and spatial correlations on the end-to-end performance of a tagged traffic stream which can be due to background traffic or partial commonality in the routing path. They proposed a binary queueing activity indicator to provide for a simple mechanism to capture these correlations. Montagna et al. [64] studied and developed an analytical approach for calculating the inter-departure time distribution of multimedia RTP traffic mixed to disturbing background traffic

across a single router. Both periodic and on-off binary Markov sources were considered in their work.

This Chapter is organized as follows. In section 5.3, we define the tandem queueing network under study. In section 5.4 we describe an exact and efficient algorithm for calculating the end-to-end delay in the tandem queueing network assuming no background arrivals. In section 5.5 we calculate the required bandwidth as a function of the number of homogeneous video streams so that the end-to-end percentile delay remains the same. In section 5.6, we extend the above algorithm to include background traffic in the form of a trace and use it to present a brief analysis on jitter and percentile of a tagged stream in section 5.7. Conclusions and future directions are presented in section 5.8

5.3 The Tandem Queueing Network under Study

We now proceed to describe an exact and efficient algorithm for calculating the end-to-end delay in a tandem queueing network consisting of N nodes, $N \geq 1$, as shown in Figure 5.1. The tagged arrival process is characterized by a video trace of a single video stream or a multiplexed set of video streams. Each node in the tandem queueing network is also fed with a background traffic that is also described by a video trace of a single stream or multiplexed streams. A video trace is a sequence of IP packets identified by the time that the packet arrives and its length in bits. The background packets that arrive at each node depart from the network after they receive service. That is, each node i has its own local background traffic that does not propagate through one or more nodes downstream from i after they complete their service. The service time at each node is constant equal to the time it takes to transmit out a packet. This can be easily obtained given that we know the speed of the transmitter. Each packet in the tagged stream keeps its length throughout the tandem queueing network and all tagged and background packets are served in a FIFO manner at each node. Also, each node i in the network may have a different transmission speed characterized by μ_i .

Figure 5.1 Tandem Queueing Network under study

We first describe the algorithm for a tandem network assuming no background arrivals. Then in section 5.5 we extend the algorithm to include background arrivals as well.

5.4 The Proposed Algorithm: No Background Arrivals

We first describe the algorithm for a single node. Then, we extend it to a tandem network with N nodes, $N \geq 1$. The total wait time *WaitTime* of a packet in a node is composed of two components:

1. Time spent waiting in the queue (if server is busy upon arrival) and
2. Time spent in service

As we mentioned earlier, for each packet we know its length in bits, and if the service rate of the node is μ (bits per second), then the service time is length/μ . The time spent waiting in the queue depends upon whether a packet arrives to find the server empty or not. If the server is idle upon arrival of a packet p , i.e., there are no packets in front of p , then the waiting time in the queue is 0. If there is one or more packets in front of p , then its waiting time is the time elapsed from the instance p arrived to the instance that the packet in front of p completes its service. Hence we have the following two cases for the total waiting time:

- If server is free, then $\text{WaitTime} = \text{length}/\mu$
- If server is busy, then $\text{WaitTime} = (\text{ServiceCompletionTime} - \text{PacktArTime}) + \text{length}/\mu$

The *ServiceCompletionTime* is defined as the time instance at which the last packet waiting in the queue departs. This variable is updated each time a new packet joins the queue. The time instance of the arrival of a new packet is indicated by *PacketArTime*.

If the server was free upon arrival of packet p , that is, $ServiceCompletionTime < PacketArTime$, then the new value for *ServiceCompletionTime* is simply the service time of the packet (i.e., $length_p/\mu$) added to the current value. If the server is already busy (which means that $(ServiceCompletionTime > PacketArTime)$) then we need to add the service time of the packet to the existing value of *ServiceCompletionTime*. This is summarized as follows, where the subscript p refers to the packet that just arrived:

If server is free,
 $WaitTime_p = length_p/\mu$
 $ServiceCompletionTime = PacketArTime_p + length_p/\mu$
 If server is busy,
 $WaitTime_p = (ServiceCompletionTime - PacketArTime_p) + length_p/\mu$
 $ServiceCompletionTime = ServiceCompletionTime + length_p/\mu$

This can be easily translated into code, and below we present the excerpt from the java implementation of the algorithm:

```
curTm=packetArTm;
if(serverFreeTm<curTm)//server is idle
{
    waitTm=packetLen/meu;
    serverFreeTm=curTm+packetLen/meu;
}
else//server is busy and/or there is a queue build up
{
    waitTm=(serverFreeTm-curTm)+packetLen/meu;
    serverFreeTm+=packetLen/meu;
}
```

The algorithm can be easily extended to multiple nodes in tandem network. For this, we need to make the following changes/extensions to the basic algorithm:

1. There is a *ServiceCompletionTime_i* for each node i .

2. The WaitTime_p of a packet is obtained by adding the $\text{WaitTime}_{p,i}$ of packet p at each node i as it moves through the tandem network.
3. The $\text{PacketArTime}_{p,i}$ of a packet p at node i is updated as it moves through the network in a way such that the arrival time at node i would be equal to the departure time from node $i - 1$.

The algorithm is as follows for packet p at node i :

If server _{i} is free upon arrival of packet p ,

$$\text{WaitTime}_{p,i} = \text{length}_p / \mu_i$$

$$\text{ServiceCompletionTime}_i = \text{PacketArTime}_{p,i} + \text{length}_p / \mu_i$$

If server _{i} is busy,

$$\text{WaitTime}_{p,i} = (\text{ServiceCompletionTime}_i - \text{PacketArTime}_{p,i}) + \text{length}_p / \mu_i$$

$$\text{ServiceCompletionTime}_i = \text{ServiceCompletionTime}_i + \text{length}_p / \mu_i$$

$$\text{TotalWaitTime}_p = \text{TotalWaitTime}_p + \text{WaitTime}_{p,i}$$

$$\text{PacketArTime}_{p,i+1} = \text{PacketArTime}_p + \text{WaitTime}_{p,i}$$

In summary, the arrival times of the packets from the trace are used to calculate their total waiting time in node 1 and their departure times from node 1. This is easy to do because the service times are deterministic, equal to length/μ_1 , where μ_1 is given. Then the arrival time of each packet to queue 2 is set equal to the departure time from queue 1, and the algorithm repeats. So if we want to simulate N nodes, then we do N rounds of the algorithm.

An interesting case arises when all nodes have the transmission rates, that is, $\mu_i = \mu$, $i=1,2,\dots,N$. In this case, the delay incurred by a packet in node 2 is the same as the delay incurred in node 3, 4, ..., N . That is, if the packet delay in node 2 is d , then the packet delay from node 2 to node N is $(N-1)d$. Hence, we only need to run the algorithm for just the first two nodes.

In order to explain this, we consider two packets, p_1 and p_2 , with p_1 being in front of p_2 , and we examine the second node at the instance when p_2 finishes its service at node 1 and moves to node 2. The following two cases are possible when p_2 joins the second node.

1. Length of $p_1 \leq$ Length of p_2 . In this case, p_2 will find no one in node 2 and will have a zero wait time, i.e., it will immediately be scheduled for service. This is because by the time p_2 completes its service at node 1, p_1 would have already completed its service at node 2 and moved on to node 3. Not only at node 2, but p_2 will also have a zero waiting time in all the other nodes because each time it reaches a node p_1 would have already moved on to the next node 4.
2. Length of $p_1 \geq$ Length of p_2 . In this case, when p_2 reaches node 2, p_1 is still in service (equal to length/μ) and p_2 will have to wait for the remainder of the time p_1 is in service. Let us call this time r . So the waiting time of p_2 at node 2 is r . This situation repeats when p_2 finishes its service at node 2 and moves to node 3, and so on. Packet p_2 will wait r time in each of the subsequent queues.

In both cases, the delay that p_2 experiences in node 2 is the same as the delay it experiences in all the subsequent nodes. So, the total packet delay from node 2 to node N is $(N-1)d$, where d is either 0 or r as described above. Hence,

$$\text{TotalWaitTime}_p = \text{WaitTime}_{p,1} + (N-1) \times \text{WaitTime}_{p,2}$$

5.5 Bandwidth Requirement for Homogeneous Flows

In this section, we used the above algorithm to calculate the required bandwidth as a function of the number of homogeneous video streams so that the end-to-end percentile delay remains the same. The video streams follow the same route depicted by a 10-node tandem queueing network, where all nodes have the same transmission time. We used the same three traces, CISCO Telepresence, WebEx and IPTV, used in the Chapter 4.

5.5.1 Telepresence

The Cisco Telepresence trace has a lag-1 autocorrelation $\rho=-0.1383$, $c^2=4.40$ and an arrival rate $\lambda=7.34$ Mbps. The distribution of the packet sizes and the arrival rate (Mbps) are given in Figure 4.4. We observe that the packet size varied from 60 B to 1260 B, and the arrival bit rate is almost constant. The almost constant arrival rate indicates that there are no scene changes in the video, which is normal for teleconferences.

Since the same bandwidth should be allocated to each node along the path of the video streams, we have that all nodes in the tandem queueing network have the same service rate. Consequently, we used the simplified algorithm presented above where we only analyze the first node, in order to calculate the end-to-end delay, and subsequently its 95th percentile. This algorithm was embedded in a simple search procedure (also used in the previous Chapters) for calculating the bandwidth that should be allocated so that the 95th percentile is 50 msec.

We obtained results by varying the number n of multiplexed traces from 1 to 30. Each trace was started at a random time uniformly distributed within the window of 0 to 25 seconds, so that to avoid temporal synchronization of the traces. The algorithm presented above calculates the end-to-end delay for each packet in a trace, which may be a single trace or a superposition of a number of traces. Obviously, the longer is the trace the better are the results, as in an event-based simulation. In view of this, we looped around each individual trace as many times as required so that half the width of the confidence interval was about 10% of the estimated end-to-end percentile. However, as the service time is deterministic, we found out that looping around three times is the maximum required to get to the required confidence interval. The confidence interval was obtained using that batch means method.

If a single trace is looped around infinite times to generate an infinite stream of packets, and then we run n (where $n=1 \dots 30$) of these streams concurrently, we will get a steady stream of packets after an initial warm up period. The arrival rate of this steady stream

will be indeed n times the arrival rate of a single stream and we would expect c^2 to either go down or remain constant as n increases. Such a combined stream is shown in Figure 5.2.

Figure 5.2 Infinite Looping of Trace to create Background Traffic

Table 5.1 gives some statistics regarding the n multiplexed traces, for $n=1, 10, 20$, and 30. We observe that the autocorrelation is zero. Also, the burstiness remains constant as the number of streams increase. This makes sense because the arrival rate of the Telepresence trace is almost constant, hence multiplexing it multiple times does not create any additional burstiness. Similarly, if n streams are combined together, we would expect the arrival rate of the combined stream to be n times the arrival rate of the original stream. These statistics are reported in Table 5.1.

Table 5.1 Statistics of Multiplexed Telepresence Traces

Number of Streams	Arrival Rate (λ) Mbps	Autocorrelation (ρ)	Burstiness (c^2)
1	7.34	-0.1383	4.40
10	73.2	-0.0938	3.15
20	146	-0.0477	3.60
30	220	-0.0582	3.43

The burstiness and autocorrelation characteristics of the combined homogeneous Telepresence streams are shown in Figure 5.3.

Figure 5.3 Burstiness (left) and Autocorrelation (right) of combined Telepresence streams

The results obtained, labeled “required bandwidth”, are given in Figure 5.4, Figure 5.5 and Figure 5.6. (The confidence intervals were not plotted since they are not discernible in the graphs.) In addition, we plotted the average arrival rate of the multiplexed stream, labeled “average bandwidth”, and the bandwidth obtained by multiplying the required bandwidth of a single stream times the number of streams, labeled “no statistical gain”. The latter measure is the bandwidth required assuming that no statistical gain is obtained by multiplexing n streams.

The three figures correspond to the three experimental conditions where the end-to-end percentile delay was fixed to be 30ms, 50ms and 70ms respectively. We note some interesting properties. First, the average bandwidth required remains the same for all these three cases. This is indeed the case as the average bandwidth required is the least amount of bandwidth required to keep the system stable and it solely depends on the average arrival rate. Second, as the end-to-end percentile delay constraint is relaxed (from 30 to 70) we see that the required bandwidth curve becomes closer and closer to the average bandwidth curve. Again this is intuitive as in order to satisfy tighter delay bounds (smaller values) we would

require additional bandwidth than the average bandwidth. Similarly, if we keep relaxing the delay bounds, lesser bandwidth will be required to fulfill them.

As can be seen, the required bandwidth is a linear function of the number of streams n . It is also very close the case where no statistical gain is achieved, which is not surprising, since the transmission rate is almost constant due to lack of scene changes. The differences between the “required bandwidth” curve and the “no statistical gain” are plotted as “bandwidth difference” in a separate graph to highlight the linear difference between curves. We would like to mention here that as the required bandwidth is calculated iteratively, we decided on a single bandwidth step size for all the experiments. Hence, we see smooth difference curves for higher bandwidth values. As the ratio of the step size to required bandwidth increases, the curves smooth out with limiting case being straight lines.

Figure 5.4 Bandwidth Requirement for Fixed 95th Percentile Delay of 30msec (left). Difference between No Statistical Gain and Required BW (right)

Figure 5.5 Bandwidth Requirement for Fixed 95th Percentile Delay of 50msec (left). Difference between No Statistical Gain and Required BW (right)

Figure 5.6 Bandwidth Requirement for Fixed 95th Percentile Delay of 70msec (left). Difference between No Statistical Gain and Required BW (right)

5.5.2 WebEx

The WebEx trace has a lag-1 autocorrelation $\rho=0.0486$, $c^2=49.5135$ and an arrival rate $\lambda=0.309$ Mbps. The distribution of the packet sizes and the arrival rate (Mbps) are given in Figure 4.8

We increased the number of concurrent streams from 1 to 30 where each stream is assigned a random starting point uniformly distributed within the window of 0 to 300

seconds. This helps in preventing the temporal synchronization of the streams. Table 5.2 gives some statistics for $n=1, 10, 20, 30$. The autocorrelation increases slightly and burstiness decreases as the number of streams increases. It makes sense for the autocorrelation to increase as we are multiplexing homogeneous flows together. Also, the arrival process of the original WebEx trace is highly bursty (see Figure 4.8). So the burstiness of the superposed stream decreases as the number of streams increases and tends towards a constant value. This is intuitive as in general we would expect that the more streams you superpose, the lower the c^2 . The discussion on arrival rate is the same as presented in section 5.5.1

Table 5.2 Statistics of Resultant Trace as multiple WebEx traces get multiplexed together

Number of Streams	Arrival Rate (λ) Mbps	Autocorrelation (ρ)	Burstiness (c^2)
1	0.3	0.0486	49.51
10	3.07	0.1340	11.66
20	6.18	0.1721	9.13
30	9.26	0.1943	8.01

The burstiness and autocorrelation characteristics of the combined homogeneous WebEx streams are shown in Figure 5.7

Figure 5.7 Burstiness (left) and Autocorrelation (right) of combined WebEx streams

The results obtained, labeled “required bandwidth”, are given in Figure 5.8, Figure 5.9 and Figure 5.10. (The confidence intervals were not plotted since they are not discernible in the graphs.) In addition, we plotted the average arrival rate of the multiplexed stream, labeled “average bandwidth”, and the bandwidth obtained by multiplying the required bandwidth of a single stream times the number of streams, labeled “no statistical gain”. The latter measure is the bandwidth required assuming that no statistical gain is obtained by multiplexing n streams.

The three figures correspond to the three experimental conditions where the end-to-end percentile delay was fixed to be 30ms, 50ms and 70ms respectively. We note some interesting properties. First, the average bandwidth required remains the same for all these three cases. This is indeed the case as the average bandwidth required is the least amount of bandwidth required to keep the system stable and it solely depends on the average arrival rate. Second, as the end-to-end percentile delay constraint is relaxed (from 30 to 70) we see that the required bandwidth curve becomes closer and closer to the average bandwidth curve. Again this is intuitive as in order to satisfy tighter delay bounds (smaller values) we would require additional bandwidth than the average bandwidth. Similarly, if we keep relaxing the delay bounds, lesser bandwidth will be required to fulfill them.

As can be seen, the required bandwidth is a linear function of the number of streams n . The differences between the “required bandwidth” curve and the “no statistical gain” are plotted as “bandwidth difference” in a separate graph to highlight the linear difference between curves. We see a huge gain in bandwidth. We can deduce from these experiments that bandwidth gain increases as burstiness of original traffic increases and as arrival rate of the original traffic decreases.

Figure 5.8 Bandwidth Requirement for Fixed 95th Percentile Delay of 30msec (left). Difference between No statistical Gain and Required BW (right)

Figure 5.9 Bandwidth Requirement for Fixed 95th Percentile Delay of 50msec (left). Difference between No statistical Gain and Required BW (right)

Figure 5.10 Bandwidth Requirement for Fixed 95th Percentile Delay of 70msec (left). Difference between No statistical Gain and Required BW (right)

5.5.3 IPTV

The IPTV trace has a lag-1 autocorrelation $\rho=0.0579$, $c^2=3.6558$ and an arrival rate $\lambda=8.83$ Mbps. The distribution of the packet sizes and the arrival rate (Mbps) are given in Figure 4.6. We observe that the trace had a small number of signaling packets, and the bulk of the traffic was video packets of size 1324B.

Table 5.3 Statistics of Resultant Trace as multiple IPTV traces get multiplexed together

Number of Streams	Arrival Rate (λ) Mbps	Autocorrelation (ρ)	Burstiness (c^2)
1	8.83	0.0579	3.65
10	88.3	-0.1686	2.54
20	177	-0.1011	2.36
30	265	-0.1004	3.31

The burstiness and autocorrelation characteristics of the combined homogeneous IPTV streams are shown in Figure 5.11

Figure 5.11 Burstiness (left) and Autocorrelation (right) of combined IPTV streams

We increased the number of concurrent streams from 1 to 30 where each stream is assigned a random starting point within the window of 0 to 15 seconds. This helps in preventing the temporal synchronization of the streams. Table 5.3 gives statistics for $n=1, 10, 20$, and 30 multiplexed streams. The autocorrelation is very low and burstiness remains constant. The discussion on burstiness and arrival rate is the same as presented in section 5.5.1

The results obtained, labeled “required bandwidth”, are given in Figure 5.12, Figure 5.13 and Figure 5.14. (The confidence intervals were not plotted since they are not discernible in the graphs.) In addition, we plotted the average arrival rate of the multiplexed stream, labeled “average bandwidth”, and the bandwidth obtained by multiplying the required bandwidth of a single stream times the number of streams, labeled “no statistical gain”. The latter measure is the bandwidth required assuming that no statistical gain is obtained by multiplexing n streams.

The three figures correspond to the three experimental conditions where the end-to-end percentile delay was fixed to be 30ms, 50ms and 70ms respectively. We note similar interesting properties as above. First, the average bandwidth required remains the same for all these three cases. This is indeed the case as the average bandwidth required is the least

amount of bandwidth required to keep the system stable and it solely depends on the average arrival rate. Second, as the end-to-end percentile delay constraint is relaxed (from 30 to 70) we see that the required bandwidth curve becomes closer and closer to the average bandwidth curve. Again this is intuitive as in order to satisfy tighter delay bounds (smaller values) we would require additional bandwidth than the average bandwidth. Similarly, if we keep relaxing the delay bounds, lesser bandwidth will be required to fulfill them.

As can be seen, the required bandwidth is a linear function of the number of streams n . It is also very close the case where no statistical gain is achieved. The differences between the “required bandwidth” curve and the “no statistical gain” are plotted as “bandwidth difference” in a separate graph to highlight the linear difference between curves. We would like to mention here that as the required bandwidth is calculated iteratively, we decided on a single bandwidth step size for all the experiments. Hence, we see smooth difference curves for higher bandwidth values. As the ratio of the step size to required bandwidth increases, the curves smooth out with limiting case being straight lines.

Figure 5.12 Bandwidth Requirement for Fixed 95th Percentile Delay of 1 sec (left). Difference between No statistical Gain and Required BW (right)

Figure 5.13 Bandwidth Requirement for Fixed 95th Percentile Delay of 2 sec (left). Difference between No statistical Gain and Required BW (right)

Figure 5.14 Bandwidth Requirement for Fixed 95th Percentile Delay of 3 sec (left). Difference between No statistical Gain and Required BW (right)

5.5.4 Linear Bandwidth Gain

The linearity behavior of bandwidth gain came as a surprise as it was expected the gain to increase exponentially as the number of homogeneous streams increases. To make

sure that this is not an aberration of the traces that we are using, we devised an independent experiment to validate this behavior.

An Interrupted Poisson Process (IPP) is characterized by an exponentially distributed ON state, an exponentially distributed OFF state and arrivals which occur only during the ON state with an exponential rate. Using the parameters of an IPP one can easily construct an $(n+1)$ -state MMPP which is the exact representation of the aggregation of n homogeneous IPP streams. This $(n+1)$ -state MMPP can then be used as an arrival process in analytical and simulation performance models of IP networks to calculate various metrics, such as end-to-end delay and queue length distributions.

IPP modulates between two exponentially distributed states, an ON and an OFF state. The transition rates for the ON and OFF states are respectively α and β , see Figure 5.15. While in the ON state, the process generates Poisson arrivals with rate λ whereas in the OFF state there are no arrivals.

Figure 5.15 Markov Chain of an IPP

The aggregate of n homogeneous IPPs (with parameters α , β and λ) is an MMPP with $n+1$ states. The underlying Markov Chain of this MMPP- $(n+1)$ process is given in Figure 5.16.

Figure 5.16 Markov Chain of an $(n+1)$ state MMPP representing n aggregate IPPs

The parameter values chosen for the IPP stream are set equal to the parameter values of a VoIP call. A VoIP has, on average, 400 ms of ON period, 600 ms of OFF period [6], and the number of packets generated per second during the ON period is 50. A voice payload of 160 bytes is carried in a separate IP packet with a 40-byte IP/UDP/RTP header. Thus, we have $\alpha=2.5$, $\beta=1.667$ and $\lambda=50$.

Figure 5.17 Bandwidth Requirement for Fixed 95th Percentile Delay of 50 msec

Figure 5.18 Difference between No statistical Gain and Required BW

As can be seen in Figure 5.17, the required bandwidth is a linear function of the number of streams n . The differences between the “required bandwidth” curve and the “no statistical gain” are plotted as “bandwidth difference” in Figure 5.18 to highlight the linear difference between curves. Again here, like in the WebEx trace, we see a huge gain in bandwidth and for the same reason too. Intuitively, an IPP is bursty because the arrivals are batched together during the ON period and no arrivals occur during the OFF period. An IPP is a close approximation to the bursty arrival process of a voice call. Hence like the results for the WebEx trace, we see a huge gain in bandwidth.

The linearity of the above results for the three traces has been independently confirmed by Lone [65]. This behavior may be explained by an algorithmic asymptotic analysis. For this, we will make use of the definition of the big-theta Θ which implies asymptotic “equality”. Before we proceed we need to define two other terms, big-oh $\mathcal{O}$ and big-omega Ω . The following definitions and Figure 5.19 is taken from the book Introduction to Algorithms by Cormen [66].

Definition of big-oh: Let f and g be functions from the set of integers (or the set of real numbers) to the set of real numbers. Then $f(n)$ is said to be $\mathbf{O}(g(n))$, which is read as $f(n)$ is big-oh of $g(n)$, if and only if there are constants C and n_0 such that

$$|f(n)| \leq C |g(n)|, \text{ whenever } n > n_0$$

Big-oh is concerned with the "less than or equal to" relation between functions for large values of the variable. It is also possible to consider the "greater than or equal to" relation and "equal to" relation in a similar way. Big-Omega is for the former and big-theta is for the latter.

Definition of big-omega: Let f and g be functions from the set of integers (or the set of real numbers) to the set of real numbers. Then $f(n)$ is said to be $\mathbf{\Omega}(g(n))$, which is read as $f(n)$ is big-omega of $g(n)$, if there are constants C and n_0 such that

$$|f(n)| \geq C |g(n)|, \text{ whenever } n > n_0$$

Definition of big-theta: Let f and g be functions from the set of integers (or the set of real numbers) to the set of real numbers. Then $f(n)$ is said to be $\mathbf{\Theta}(g(n))$, which is read as $f(n)$ is big-theta of $g(n)$, if $f(n)$ is $\mathbf{O}(g(n))$ and $\mathbf{\Omega}(g(n))$. We also say that $f(n)$ is of order $g(n)$ or that both f and g are of the same order. $f(n) = \mathbf{\Theta}(g(n))$ implies: $f(n) \sim g(n)$. Formally,

$$\Theta(g(n)) = \{f(n) : \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}.$$

This is represented pictorially in Figure 5.19, i.e., $g(n)$ is an asymptotically tight bound for $f(n)$.

Figure 5.19 Pictorial Representation of big-theta Θ

We know that a straight line is defined by the equation $f(n) = mn + b$, where m is the slope or gradient of the line and b is the Y intercept. In all three Figures (Figure 5.4, Figure 5.8 and Figure 5.13), the required bandwidth curve is bounded by the average bandwidth curve from below and the no statistical gain curve from above. Let us name the required bandwidth curve $f(n)$, where n is the number of streams and let $g(n) = n$. Hence we can define the average bandwidth curve by the straight line $g_1(n) = m_1n$ since its Y intercept is 0. Similarly, the no statistical gain curve can be defined by the straight line $g_2(n) = m_2n$. Now we have all the components of the big-theta definition, i.e.,

$$0 \leq m_1g(n) \leq f(n) \leq m_2g(n) \text{ for all } n \geq 0$$

Hence by definition of big-theta, $f(n) = \Theta(g(n))$ or $f(n) \sim g(n)$. As $g(n)$ is a linear line, hence $f(n)$ will also be a linear line.

In simpler words, if a function $f(n)$ is bounded from above by a linear curve and also from the below by a linear curve, then it itself behaves in a linear fashion. This is in agreement with the curves above as the required bandwidth curve is bounded from below with a linear average bandwidth curve and on the above with the linear no statistical gain requirement. The required bandwidth will remain bounded by these two curves and hence behaves as a linear curve asymptotically.

5.6 Analysis of the Tandem Queueing Network with Background Traffic

The basic algorithm presented in section 5.4 can be easily extended to include background traffic in the form of a trace. It is immaterial to the algorithm whether the background traffic is a single stream or a superposition of several streams. The only requirement is that the trace contains the arrival time of each packet and its packet length. Also, the background trace may be node-dependent. That is, it may vary from node to node. Finally, the background traffic to each node does not propagate to downstream nodes. As shown in Figure 5.1, a packet of the background process that joins node i departs from the queueing network when it completes its service at node i .

The algorithm follows the same basic steps as before. The only difference lies in the selection of the next packet. In the previous algorithm, it was simply the next packet in the trace. In this case, we have to make a decision between the next packet of the trace and the next packet of the background trace. This can be resolved by simply comparing the arrival time of the next packet in the tagged trace ($\text{PacketArTime}_{\text{Tagged}}$) and the next packet in the background trace ($\text{PacketArTime}_{\text{Bkgrd}}$). The tagged trace is the one that traverses the entire queueing network and for which we want to calculate its end-to-end delay.

```

If ( $\text{PacketArTime}_{\text{Tagged}} < \text{PacketArTime}_{\text{Bkgrd},i}$ )
  If server $i$  is free,
     $\text{WaitTime}_{p,i} = \text{length}_p / \mu_i$ 
     $\text{ServiceCompletionTime}_i = \text{PacketArTime}_p + \text{length}_p / \mu_i$ 
  If server $i$  is busy,
     $\text{WaitTime}_{p,i} = (\text{ServiceCompletionTime}_i - \text{PacketArTime}_p) + \text{length}_p / \mu_i$ 
     $\text{ServiceCompletionTime}_i = \text{ServiceCompletionTime}_i + \text{length}_p / \mu_i$ 
   $\text{TotalWaitTime}_p = \text{TotalWaitTime}_p + \text{WaitTime}_{p,i}$ 
   $\text{PacketArTime}_p = \text{PacketArTime}_p + \text{WaitTime}_{p,i}$ 
Else If ( $\text{PacketArTime}_{\text{Tagged}} > \text{PacketArTime}_{\text{Bkgrd},i}$ )
  If server $i$  is free,

```

$$\text{ServiceCompletionTime}_i = \text{PacketArTime}_{\text{Bkgrd},i} + \text{length}_{\text{Bkgrd},i}/\mu_i$$

If server_i is busy,

$$\text{ServiceCompletionTime}_i = \text{ServiceCompletionTime}_i + \text{length}_{\text{Bkgrd},i}/\mu_i$$

The algorithm is quite efficient on time. Table 5.4 gives a comparison between the proposed algorithm and a simulation program for a 10-node tandem queueing network. The simulation is event-based whereby the program keeps track of the arrival and departure events and manages the event list. We ran both versions for the three types of traces till half the width of the confidence interval is within 10% of the estimated 95th percentile of the end-to-end delay. In Table 5.4, we list the CPU time (msec) required by each of the two programs to calculate the 95th percentile of the end-to-end delay, for each of the three traces and for varying number of background streams. The number next to the trace name is the number of packets that needed to be simulated/run in order to get to the required confidence interval.

Table 5.4 Simulation vs Algorithm CPU Time

	Telepresence (45K)		WebEx (65K)		IPTV (20K)	
Number of background traces	Simulation CPU Time (msec)	Algorithm CPU Time (msec)	Simulation CPU Time (msec)	Algorithm CPU Time (msec)	Simulation CPU Time (msec)	Algorithm CPU Time (msec)
1	1234	246	2414	400	254	186
10	4293	746	7723	1295	535	264
20	10036	1656	9625	1374	1670	614
30	18560	2320	13311	1479	3247	947

5.7 Jitter and Delay Percentile Analysis

We used the above algorithm to see how the 95th percentile delay and jitter of a tagged stream changes end-to-end and at each individual node of the tandem network when we mix the tagged stream with different intensities of local background traffic.

5.7.1 Jitter

Jitter is a variation in packet transit delay caused by queueing, contention and serialization effects on the path through the network. In general, higher levels of jitter are more likely to occur on either slow or heavily congested links. Jitter is a notion and not a measure. In view of this, various measures have been proposed and used for jitter. We will use the commonly used measure of mean packet-to-packet delay variation (MPPDV) in our work [67]. Let t_i and t_{i+1} be the delay of two successive packets i and $i+1$, then the packet to packet delay variation is $|t_{i+1}-t_i|$. The mean packet to packet delay variation is the average of all differences $|t_{i+1}-t_i|$ of all packets.

5.7.2 Results

We consider a three-node tandem queueing network with background arrivals to each node, and use Telepresence and WebEx traces for the study. The IPTV trace was not considered because IPTV does not have stringent end-to-end and jitter requirements. A three-node network is selected for this study so that the effects of loading one node at a time (first, second or third) could be easily discerned. In all our experiments, our tagged traffic remains the same while we vary the amount of background traffic at each node. The background traffic is generated by multiplexing the tagged trace multiple times, with a random starting point for each trace, so that to avoid temporal synchronization. We have calculated results for each of the two traces for the following three scenarios:

1. The background traffic on all three nodes is the same to start with, and it increases by the same amount.
2. The background traffic in the first and last node is the same and it remains constant. The second node starts with a light background load and then its load is slowly increased to become the heaviest utilized node of the three.
3. The background traffic in the first and the second node is the same and it remains constant. The last node starts a light background load and its load is slowly increased to become the heaviest utilized node of the three.

5.7.3 Telepresence

In the first scenario, the load on all three nodes increases by the same intensity. The number of streams multiplexed to make up the background traffic is shown on the horizontal axis of all the Figures in this section. As shown in Figure 5.20, the first node is the dominant node and contributes the most to the 95th percentile delay. This is consistent with the findings of the previous chapters. The first node dominance is also visible in the jitter graph, but here, interestingly, we see that the growth trend of node 1 is slower than that of node 2 and 3. This means that as the background traffic increases, node 2 and 3 become the major contributing nodes to the over all jitter.

Figure 5.20 The 95th percentile delay (left) and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3 when the background load on all three nodes increases proportionately

In the second scenario, the background traffic in the first and last node is the same, a superposition of 10 telepresence streams, and it remains constant as the background traffic load to node 2 increases. Hence, we see that the percentile delay and jitter of both node 1 and 3 remain constant in Figure 5.22. We observe that the first node is dominant on both percentile delay and jitter in the beginning. The second node starts lightly loaded and its load is slowly increased (as indicated on the horizontal axis) to become the heaviest utilized node of the three. As expected, it starts by being the least significant contributing node to the end-to-end delay and jitter, and slowly increases until it becomes the most dominant node, after

the node 2 curves cross the curves of the other nodes. Similar results are obtained by keeping the first and second node constant and increasing the background load on the third node. The results are presented in Figure 5.22.

Figure 5.21 The 95th percentile delay (left) and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 3 while the background load on node 2 increases.

Figure 5.22 The 95th percentile delay (left), and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 2 while the background load on node 3 increases.

5.7.4 WebEx

In the first scenario, the load on all three nodes increases by the same intensity. The number of streams multiplexed to make the background traffic are shown on the horizontal

axis of all the figures in the section. As shown in Figure 5.23, the first node is the dominant node in the 95th percentile delay. This is consistent with the findings of the previous chapters. The first node dominance is also visible in the jitter graph. However, unlike Telepresence, the increase of the per node jitter is the same in all three nodes as the background traffic increases.

Figure 5.23 The 95th percentile delay (left), and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3 when the background load on all three nodes increases proportionately

In the second scenario, the background traffic on the first and last node is the same, a mixture of 10 WebEx streams, and it remains constant as the background traffic load of node 2 increases. Hence we see that the percentile delay and jitter of both node 1 and 3 remains constant in Figure 5.24. We observe that the first node is dominant on both percentile delay and jitter in the beginning. The second node starts lightly loaded and its load is slowly increased (as indicated by the horizontal axis) to become the heaviest node of the three. As expected, it starts by being the least significant contributing node to the end-to-end delay and jitter, and slowly increases until it becomes the most dominant node, after the node 2 curves cross the curves of the other nodes. Similar results are obtained by keeping the first and second node constant and increasing the background load on the third node. The results are presented in Figure 5.25.

Figure 5.24 The 95th percentile delay (left), and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 3 while the background load on node 2 increases.

Figure 5.25 The 95th percentile delay (left) and jitter (right) calculated end-to-end, and for nodes 1, 2 and 3. Constant background traffic for nodes 1 and 2 while the background load on node 3 increases.

5.8 Bandwidth Allocation under Jitter and Percentile Delay Bounds

In the previous section we examined how the end-to-end percentile delay and the jitter reacts to the increase in the background traffic for a given service rate. It will be equally interesting to find out the complementary view, i.e., how the bandwidth requirements change with the increase in background traffic for a fixed end-to-end percentile delay or a fixed end-to-end jitter value (or both).

In this section, we first calculate the bandwidth required to satisfy a fixed end-to-end percentile delay bound and jitter bounds (separately) for increasing background load. Later we discuss how we can combine both these bounds using our algorithm so as to calculate the bandwidth requirement that can satisfy both delay and jitter bounds simultaneously. The results are presented below for each of the three types of traces, Telepresence, WebEx and IPTV.

5.8.1 Telepresence

We are considering two bounds for the current experiments, percentile delay and jitter. First we present bandwidth results when a single bound, i.e., percentile delay or jitter is in effect along with the corresponding values of the other metric of interest, i.e., jitter or percentile delay respectively.

Figure 5.26 Bandwidth and Jitter values for a 95th Percentile Delay of 50msec

Figure 5.27 Bandwidth and Percentile Delay values for Jitter of 10msec

We notice that the bandwidth requirement increases with the increase in background traffic for both fixed percentile delay and jitter. This observation leads us to using the trace algorithm in an iterative scheme. For an initial small bandwidth value, we calculate the end-to-end percentile delay and jitter values for a given flow. The initial value is chosen such that neither of the two constraints are met. Next we increase the bandwidth value by a fixed step size, and re-calculate the values. After a few iterations the less stringent requirement will be met, which in our examples is the jitter constraint. We mark that one constraint is met and keep on iterating with step size increases in bandwidth until the second constraint, which is percentile end-to-end delay in our examples, is also satisfied. We note down this value as the required bandwidth that satisfies both bounds. Next, we increase the background traffic and repeat the process.

Figure 5.28 Required bandwidth for dual bounds

Required bandwidth results are presented in Figure 5.28. The red curve indicates the bandwidth required to satisfy end-to-end jitter value of 10ms, the blue curve represents the bandwidth required to satisfy the end-to-end 95th percentile delay value of 50msec. As we can see the delay curve completely dominates the jitter curve and hence also represents the bandwidth that is required to satisfy both the bounds, as shown by black cross lines.

To see if the algorithm works in a case where one curve is not completely dominated by another, we decreased the required jitter value from 10msec to 1msec. Resultantly, the bandwidth requirement increased and intersected with the percentile requirement. This is shown in Figure 5.29. So for background traffic which consists of 10 or less flows, the average jitter constraint (red curve) dominates, and for more than 10 flows the percentile delay (blue curve) constraint dominates. We used our algorithm for these constraint values and again were able to successfully calculate the upper bound on the bandwidth requirement, i.e., the value of bandwidth that satisfies both bounds at the same time.

Figure 5.29 Required bandwidth for dual bounds

It should be noted here that the jitter bounds of 1msec is not a value of practical significance, and was chosen for the sake of verifying the correct working of the algorithm. The required jitter values in networks vary from 10-30 msec.

5.8.2 WebEx

We repeat the same experiments for WebEx traces. First we present bandwidth results when a single bound, i.e., percentile delay or jitter is in effect along with the corresponding values of the other metric of interest, i.e., jitter or percentile delay respectively.

Figure 5.30 Bandwidth and Jitter values for 95 Percentile Delay of 50msec

Figure 5.31 Bandwidth and Percentile Delay values for Average Jitter of 10msec

We notice that the bandwidth requirement increases with the increase in background traffic for both fixed percentile delay and jitter. This observation leads us to using the trace algorithm in an iterative scheme as described in the previous section.

Figure 5.32 Required bandwidth for dual bounds

Required bandwidth results are presented in Figure 5.28. The red curve indicates the bandwidth required to satisfy the end-to-end jitter value of 10ms, the blue curve represents the bandwidth required to satisfy the end-to-end 95 percentile delay value of 50msec. As we can see the delay curve completely dominates the jitter curve and hence also represents the bandwidth that is required to satisfy both the bounds, as shown by black cross lines.

To see if the algorithm works in a case where one curve is not completely dominated by another, we decreased the required average jitter value from 10 msec to 1 msec. Resultantly, the bandwidth requirement increased and intersected with the percentile requirement. This is shown in Figure 5.29. So for background traffic which consists of 10 or less flows, the average jitter constraint (red curve) dominates, and for more than 10 flows the percentile delay (blue curve) constraint dominates. We used our algorithm for these constraint values and again were able to successfully calculate the upper bound on the bandwidth requirement, i.e., the value of bandwidth that satisfies both bounds at the same time.

Figure 5.33 Required bandwidth for dual bounds

5.8.3 IPTV

We repeat the same experiments for IPTV traces. First we present bandwidth results when a single bound, i.e., percentile delay or jitter is in effect along with the corresponding values of the other metric of interest, i.e., jitter or percentile delay respectively.

Figure 5.34 Bandwidth and Jitter values for 95 Percentile Delay of 50msec

Figure 5.35 Bandwidth and Percentile Delay values for Average Jitter of 10msec

As before, the bandwidth requirement increases with the increase in background traffic for both fixed percentile delay and jitter. This observation leads us to using the trace algorithm in an iterative scheme as described in the previous section.

Figure 5.36 Required bandwidth for dual bounds

Required bandwidth results are presented in Figure 5.28. The red curve indicates the bandwidth required to satisfy the end-to-end jitter value of 10 msec, the blue curve represents the bandwidth required to satisfy the end-to-end 95th percentile delay value of 50 msec. As

we can see the delay curve completely dominates the jitter curve and hence also represents the bandwidth that is required to satisfy both the bounds, as shown by black cross lines.

To see if the algorithm works in a case where one curve is not completely dominated by another, we decreased the required average jitter value from 10 msec to 1msec. Resultantly, the bandwidth requirement increased and intersected with the percentile requirement. This is shown in Figure 5.29. So for background traffic which consists of 10 or less flows, the jitter constraint (red curve) dominates, and for more than 10 flows the percentile delay (blue curve) constraint dominates. We used our algorithm for these constraint values and again were able to successfully calculate the upper bound on the bandwidth requirement, i.e., the value of bandwidth that satisfies both bounds at the same time.

Figure 5.37 Required bandwidth for dual bounds

5.9 Conclusions

In this chapter we propose an exact and efficient algorithm for calculating any percentile of the end-to-end delay in the presence of background video traffic, assuming that

all video flows are characterized by packet traces and an arriving packet keeps its packet length in the entire queueing network. These traces may be a single stream or multiple streams. Based on this algorithm, the minimum amount of bandwidth required, so that a given percentile of the end-to-end delay is satisfied, is easily obtained using a simple search algorithm.

Chapter 6

6 Conclusion and Future Work

In this thesis, we started by doing a survey of various Quality of Service measures that have been, and are, currently in use in the IP network. In Chapter 2, we defined percentiles and discuss their statistical significance in various networking situations. We then presented an exact analytic expression for adding percentiles of random variables whose PDF is a mixture of exponentials. Specifically we have considered exponential, Erlang, hypoexponential, hyperexponential, and Coxian-2 distributions. We also discuss the applicability of percentile calculations to routing protocols, presenting Dijkstra's algorithm as an example.

In Chapter 3 we used end-to-end delay percentiles, as the Quality of Service restriction, to propose a bandwidth allocation scheme for an MPLS-enabled IP network. In other words, we are concerned with the calculation of the bandwidth to be allocated at the output port of each router along the path of a connection, so that the end-to-end delay D is less than or equal to a given target delay value T with a probability γ , i.e., $P(D \leq T) = \gamma$. We

assume that the arrival of packets follow an MMPP2 process, because it is capable of capturing burstiness and autocorrelation characteristics commonly present in network traffic while satisfying a reduced complexity. We first construct an upper and a lower bound on a given percentile of D , from which we obtain bounds of the bandwidth such that $P(D \leq T) = \gamma$, for given T and γ . These two bounds are then combined using an interpolation function to obtain an accurate estimate of the bandwidth. The main contribution of the scheme is that the upper and lower bounds are constructed by analyzing only the first queue of the tandem queueing network.

In Chapter 4, we extended the approach described in Chapter 3 to the case where the arrival process is a two-state Markov arrival processes (MAP). A MAP can represent a variety of processes that includes, as special cases, the Poisson process, the phase-type renewal processes, the MMPP and superposition of these. Later in the chapter, we also used a MAP2 to successfully approximate the packet arrival process of various video streams, such as Cisco's Telepresence, IPTV and WebEx.

We take the bandwidth allocation of video using traces a step further in Chapter 5. The algorithm described in Chapter 4 for calculating a percentile of the end-to-end delay of a video stream, cannot handle the presence of background traffic that competes with the video within the output port of a router. In view of this, we propose an exact and efficient algorithm for calculating any percentile of the end-to-end delay in the presence of background video traffic, assuming that all video flows are characterized by packet traces and an arriving packet keeps its packet length in the entire queueing network. Using the algorithm, we calculate the required bandwidth as a function of the number of homogeneous video streams so that the end-to-end percentile delay remains the same. Also we present a brief analysis on jitter and percentile of a tagged stream in the presence of background traffic of varying intensities. Later in the chapter we extend the analysis to calculate how the bandwidth requirements of a flow changes as the background traffic varies on all the nodes constituting the end-to-end path. Finally we demonstrate the applicability of the algorithm for dual bounds situations, i.e., how can we use the algorithm to calculate the bandwidth required for

a flow that needs to satisfy constraints on both percentile delay and average jitter at the same time.

The algorithm described above has a lot of potential. It presents a very simple and elegant way to calculate connection statistics in the presence of interfering background traffic, and can be used in a variety of situations to generate interesting results. Some of the future work directions that we intend to follow are listed below

1. We would like to explore how does the algorithm generalize if the queues are finite? What kind of buffer bounds can we predict or come up with using traces.
2. Also interesting would be to see how does this generalize if we want to keep track of multiple streams in a mesh topology
3. Thirdly, all the work that we have done so far has considered background traces consisting of homogeneous video flows. It is yet to be seen what happens if we have a mix of non-homogeneous videos

REFERENCES

- [1] RFC2205, "Resource ReserVation Protocol".
- [2] RFC2212, "Specification of guaranteed quality of service," IETF.
- [3] RFC2212, "Specificatoin of the controlled-load network element service," IETF.
- [4] W. Odom and M. J. Cavanaugh, *IP Telephony Self-Study: Cisco QOS Exam Certification Guide*, 2nd ed. Cisco Press, 2004.
- [5] RFC2475, "An architecture for differentiated services," IETF.
- [6] H. G. Perros, *Connection-Oriented Networks: SONET/SDH, ATM, MPLS, and OPTICAL NETWORKS*. Wiley, 2005.
- [7] RFC2998, "A framework for integrated services operation over DiffServ networks ," IETF.
- [8] A. Meddeb, "Internet QoS: Pieces of the Puzzle," *IEEE Communications Magazine*, pp. 86-94, Jan. 2010.
- [9] X. Xiao, "Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective," *Morgan Kaufmann Series in Networking, Elsevier*, Sep. 2008.
- [10] A. Mohammed, E. Jones, and M. A. Vouk, "DiffServ experiments: analysis of the premium service over the Alcatel-NCSU Internet2 testbed," in *3rd European Conference on Universal Multiservice Networks*, 2002, pp. 124-130.
- [11] RFC3031, "Multiprotocol label switching architecture," IETF.
- [12] RFC3270, "Multi-protocol label switching (MPLS) support of differentiated services," IETF.
- [13] RFC2702, "Requirements for traffic engineering over MPLS," IETF.
- [14] C. W. Paper. (2009) CISCO. [Online].
http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6557/ps6608/whitepaper_c11-551235.html

- [15] H. S. Kim and N. B. Shroff, "Loss probability calculations and asymptotic analysis for finite buffer multiplexers," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 755-768, 2001.
- [16] C. Villamizar and C. Song, "High performance TCP in ANSNET," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 45-60, 1994.
- [17] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *ACM SIGCOMM*, 2004, pp. 281-292.
- [18] A. Dhamdhere, H. Jiang, and C. Dovrolis, "Buffer sizing for congested Internet links," in *IEEE INFOCOM*, 2005, pp. 1072-1083.
- [19] A. Dhamdhere and C. Dovrolis, "Open issues in router buffer sizing," *ACM SIGCOMM Computer Communications Review*, vol. 36, no. 1, pp. 87-92, 2006.
- [20] M. Enachescu, Y. Gangali, A. Goel, T. Roughgarden, and N. McKeown, "Part III: Routers with very small buffers," *ACM SIGCOMM Computer Communications Review*, vol. 35, no. 3, pp. 83-90, 2005.
- [21] G. Raina, D. Towsley, and D. Wischik, "Part II: Control theory for buffer sizing," *ACM SIGCOMM Computer Communications Review*, vol. 35, no. 3, pp. 79-82, 2005.
- [22] G. Vu-Brugier, R. Stanojevic, and D. L. Shorten, "A critique of recently proposed buffer sizing strategies," *ACM SIGCOMM Computer Communications Review*, vol. 37, no. 1, pp. 43-48, 2007.
- [23] A. Lakshmikantha, R. Srikant, and C. Beck, "Impact of file arrivals and departures on buffer sizing in core routers," in *IEEE INFOCOM*, 2008, pp. 86-90.
- [24] S. Wright, "Admission Control in Multi-Service IP Networks: A Tutorial," *IEEE Comm. Surveys*, vol. 9, no. 2, pp. 72-78, 2007.
- [25] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for Bounding End-to-End Delays on an AFDX Network," in *ECRTS 06*.
- [26] R. E. Kooij, O. Osterbo, and J. C. van der Wal, "Calculating End-to-End Queuing Delay for Real-Time Services on an IP Network," in *Art-QoS 2003, LNCS 2698*, 2003, pp. 115-126.
- [27] D. D. Vleeschauwer, M. J. C. Buchli, A. V. Moffaert, and R. E. Kooij, "End-to-End Queuing Delay Assessment in Multi-service IP Networks," *Journal of Statistical Computation and Simulation*, vol. 72, no. 10, pp. 803-824, 2002.

- [28] J. Abate and W. Whitt, "Numerical Inversion of Laplace Transforms of Probability Distributions," *ORSA Journal on Computing*, vol. 7, no. 1, 1995.
- [29] P. Goyal, S. S. Lam, and H. M. Vin, "Determining End-to-End Delay Bounds In Heterogeneous Networks," in *5th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1995.
- [30] S.-N. Yeung and J. Lehoczky, "End-to-end Delay Analysis for Real-Time Networks," in *22nd IEEE Real-Time Systems Symposium*, 2001.
- [31] M. Iftikhar, B. Landfeldt, and M. Caglar, "An Analytical Model Based on G/M/1 with Self-Similar Input to Provide End-to-End QoS in 3G Networks," in *4th ACM international workshop on Mobility management and wireless access*, 2006.
- [32] K. Xiong and H. Perros, "Resource Optimization Subject to a Percentile Response Time SLA for enterprise computing," in *IEEE Globecom*, 2006.
- [33] S.-N. Yeung and J. Lehoczky, "End-to-end Delay Analysis for Real-Time Networks.," in *22nd IEEE Real-Time Systems Symposium*, 2001.
- [34] M. Lelarge, Z. Liu, and C. H. Xia, "Asymptotic Tail Distribution of End-to-End Delay in Networks of Queues with Self-Similar Cross Traffic," in *INFOCOM*, 2004.
- [35] J. G. Kreifeldt and K. Nah, "Adding and subtracting percentiles - How bad can it be?," in *Human Factors and Ergonomics Society Annual Meeting Proceedings*, 1995, pp. 301-305.
- [36] M. C. F. Program, "Inter-provider Quality of Service," white paper draft 1.1., 2006.
- [37] I.-T. R. Y.1541, "Network Performance Objectives for IP-based Services," 2005.
- [38] I. Heniger. (Personal Communication).
- [39] H. Jiang and C. Dovrolis, "Why is the Internet Bursty in Short Time Scales?," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 241-252, 2005.
- [40] D. R. Figueiredo, B. Liu, V. Misra, and D. Towsley, "On the autocorrelation structure of the TCP traffic," *Computer Networks*, vol. 40, pp. 339-361, 2002.
- [41] A. Adas, "Traffic models in broadband networks," *IEEE Communication Magazine*, vol. 35, no. 7, p. 82-89, 1997.

- [42] P. J. Kuehn, "Remainder on queueing theory for ATM networks," *Telecommunication Systems* 5, pp. 1-24, 1996.
- [43] A. T. Andersen and B. F. Nielsen, "A Markovian approach for modeling packet traffic with long-range dependence," *IEEE JSAC*, vol. 16, no. 5, p. 719-732, 1998.
- [44] A. Horvath and M. Telek, "Markovian modeling of real data traffic: Heuristic phase type and MAP fitting of heavy tailed and fractal like samples," *Performance Evaluation of Complex Systems: Techniques and Tools, IFIP Performance, LNCS Tutorial Series*, vol. 2459, pp. 405-434, 2002.
- [45] A. Riska and E. Riedel, "Long-range dependence at the disk drive level," in *3rd Conference on Quantitative Evaluation of Systems (QEST)*, 2006, pp. 41-50.
- [46] H.-W. Ferng and J.-F. Chang, "Connection-wise End-to-End Performance Analysis of Queueing Networks with MMPP Inputs," *Performance Evaluation Journal*, vol. 43, pp. 39-62, 2001.
- [47] A. Jean-Marie, Z. Liu, P. Nain, and D. Towsley, "Computational Aspects of the Workload Distribution in the MMPP/G1/1 Queue," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 5, 1998.
- [48] G. Casale, E. Z. Zhang, and E. Smirni, "Trace data characterization and fitting for Markov modeling," *Performance Evaluation*, vol. 67, no. 2, pp. 61-79, 2010.
- [49] D. M. Lucantoni, K. S. Meier-Hellstern, and M. F. Neuts, "A single-server queue with server vacations and a class of non-renewal arrival processes," *Advances in Applied Probability*, vol. 22, no. 3, pp. 676-705, 1990.
- [50] D. M. Lucantoni, "New results on the single server queue with a batch Markovian arrival process," *Communications in Statistics -Stochastic Models*, vol. 7, pp. 1-46, 1991.
- [51] S. H. Kang, Y. H. Kim, D. K. Sung, and B. D. Choi, "An application of Markovian arrival process (MAP) to modeling superposed ATM cell streams," *IEEE Transactions on Communications*, vol. 50, no. 4, pp. 633-642, 2002.
- [52] A. Heindl, K. Mitchell, and A. van de Liefvoort, "Correlation bounds for second-order MAPs with application to queueing network decomposition," *Performance Evaluation - Modelling techniques and tools for computer performance evaluation*, vol. 63, no. 6, pp. 553-577, Jun. 2006.

- [53] V. Puttasubappa, "Video Traces - Personal Communication Puttasubappa, Vishwas," 2009.
- [54] D. Stanford and W. Fischer, "The interdeparture-time distribution for each class in the Σ iMi/Gi/1 queue," *Queueing Systems*, vol. 4, no. 3, pp. 179-191, 1989.
- [55] S. Dasu, "Class dependent departure process from multiclass phase queues: Exact and approximate analyses," *European Journal of Operational Research*, vol. 108, no. 2, pp. 379-404, 1998.
- [56] W. Whitt, "The Queueing Network Analyzer," *Bell System Technical Journal*, vol. 62, no. 9, pp. 2779-2815, 1983.
- [57] J. Kumaran, K. Mitchell, and A. van de Liefvoort, "An analytic model of correlations induced in a packet stream by background traffic in IP access networks," in *Proceedings of the 19th International Teletraffic Congress*, Beijing, China, 2005, pp. 687-696.
- [58] K. Mitchell and A. van de Liefvoort, "Second-order Statistics of an Isolated Departure Stream from a Shared Buffer with Correlated Sources," in *Proceedings of the 8th International Conference on Telecommunication Systems, Modeling and Analysis*, Nashville, TN, 2000, pp. 565-574.
- [59] K. Sohraby and A. Privalov, "End-to-end jitter analysis in networks of periodic flows," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Kansas City, 1999, pp. 575-583.
- [60] G. Geleji, "On Multi-Domain QoS Routing and Rate Jitter Analysis," PhD Thesis, North Carolina State University, 2011.
- [61] Y.-C. Wang, H.-P. Lin, F.-H. Tsai, and D.-L. Cai, "Elucidating the short term loss behavior of a DMAP process in random early detection queue with background DBMAP," in *Proceedings of the 6th International Conference on Queueing Theory and Network Applications*, New York, 2011.
- [62] R. L. Ioannis and I. Stavrakakis, "Traffic Shaping of a Tagged Stream in an ATM Network: Approximate End-to-End Analysis," in *IEEE INFOCOM*, 1995, pp. 162-169.
- [63] M. Conti, E. Gregori, and I. Stavrakakis, "Large impact of temporal/spatial correlations on per-session performance measures: single and multiple node cases," *Performance Evaluation*, vol. 41, pp. 83-116, 2000.

- [64] S. Montagna, R. Cemelli, and M. Pignolo, "An Analytical Study on the Interdeparture-Time Distribution for Different Multimedia Source Models in a Packet Switched Network," *Lecture Notes in Computer Science, High Performance Computing and Communications*, vol. 3726, pp. 652-659, 2005.
- [65] Q. B. Lone, "Bandwidth Allocation for Video Streams Subject To An End-to-End Percentile Delay," MS Thesis, North Carolina State University, 2011.
- [66] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Massachusetts Institute of Technology, 2009.
- [67] VoIP Troubleshooter. [Online].
<http://www.voiptroubleshooter.com/indepth/jittersources.html>
- [68] T. Osogami, "Analysis of Multiserver Systems via Dimensionality Reduction of Markov Chains," Doctoral Dissertation CMU, 2005.
- [69] M. F. Neuts, *Structures Stochastic Matrices of M/G/1 Type and Their Applications*. New York: Marcel Dekker, 1989.
- [70] M. F. Neuts, "Models based on the Markovian arrival process," *IEICE Transactions on Communications*, vol. E75-B, p. 1255–1265, 1992.
- [71] A. Klemm, C. Lindemann, and M. Lohmann, "Traffic Modeling of IP Networks Using the Batch Markovian Arrival Process," in *12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*.
- [72] A. T. Andersen, M. F. Neuts, and B. F. Nielsen, "On the time reversal of Markovian arrival processes," *Communications in Statistics - Stochastic Models*, vol. 20, no. 2, pp. 237-260, 2004.
- [73] C. Blondia, "The N/G/1 finite capacity queue," *Communications in Statistics - Stochastic Models*, vol. 5, p. 273–294, 1989.
- [74] H. W. Lee, S. H. Cheon, E. Y. Lee, and K. C. Chae, "Workload and Waiting Time Analyses of MAP/G/1 Queue under D-policy," *Queueing Systems*, vol. 48, pp. 421-443, 2004.
- [75] H. Okamura, T. Dohi, and K. S. Trivedi, "Markovian arrival process parameter estimation with group data," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, 2009.
- [76] X. Hesselbach, C. Kolias, R. Fabregat, M. Huerta, and Y. Donoso, "Problems in Dynamic Bandwidth Allocation in Connection Oriented Networks," *Graphs and Algorithms in*

- [77] S. Asmussen and J. R. Møller, "Calculation of the Steady State Waiting Time Distribution in GI/PH/c and MAP/PH/c Queues," *Queueing Systems*, vol. 37, pp. 9-29, 2001.
- [78] S. Eum, R. Harris, and I. Atov, "A Matching Model for MAP-2 using Moments of the Counting Process," in *International Network Optimization Conference*, 2007.
- [79] P. Seeling, M. Reisslein, and B. Kulapala, "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial," *Communications Surveys & Tutorials, IEEE*, vol. 6, no. 3, pp. 58-78, 2004.
- [80] P. Seeling and M. Reisslein, "Video Transport Evaluation With H.264 Video Traces," *Communications Surveys & Tutorials, IEEE*, vol. PP, no. 99, pp. 1-24, 2011.
- [81] L.-S. Chou and C.-S. Chang, "Experiments of the theory of effective bandwidth for Markov sources and video traces," *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2, no. 2, pp. 497-504, 1996.
- [82] D. E. Wrege, E. W. Knightly, H. Zhang, and J. Liebeherr, "Deterministic delay bounds for VBR video in packet-switching networks: fundamental limits and practical trade-offs," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 352-362, 1996.
- [83] B. Anjum, H. Perros, X. Mountrouidou, and K. Kontovasilis, "Bandwidth Allocation under End-to-End Percentile Delay Bounds," *International Journal of Network Management, Wiley*, vol. 21, no. 6, pp. 536-547, 2011.