Adaptive Resource Management in Label-Switched Networks

by

Wentao Shen

B.Eng.(Electrical Engineering)
Huangzhong University of Science and Technology
Wuhan, China

A thesis Submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
Faculty of Engineering

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, K1S 5B6, Canada

May 27, 2002

©2002, Wentao Shen

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-79717-1

Canada

The undersigned recommend the Faculty of Graduate
Studies and Research the acceptance of the thesis

Adaptive Resource Management in Label-Switched Networks

Submitted by Wentao Shen, B.Eng.
in partial fulfillment of the requirements for the degree of
Master of Applied Science

Professor Mihail Devetsikiotis

Thesis Supervisor

A handwritten signature in black ink, consisting of a large loop followed by a series of smaller loops and a long horizontal stroke.

Professor R.A. Goubran

Chair, Department of Systems and Computer Engineering

A handwritten signature in black ink, starting with a large 'G' and ending with a long horizontal stroke.

Abstract

The timeliness of adaptive resource allocation (i.e. *self-sizing*) requires an on-line measurement technique to report real-time traffic load as well as a fast analysis and synthesis algorithm to process these fresh data in large-scale *self-sizing* networks.

In the thesis, we realize one approach to measure effective bandwidth of traffic without any *a priori* traffic knowledge, but find it slow and non-free of potential numerical overflow in the case of *on-line* mode, hence propose one Double Recursive algorithm with Double Moving Windows (DRDMW) to overcome these two drawbacks to implement on-line EB measurements and justify the proposed DRDMW algorithm via simulation.

Also, we notice that the performance of the *simulated annealing* algorithm serving as the fast analysis and synthesis algorithm deteriorates with the scale-up of network; hence we propose a hierarchical framework to improve its performance in the large-scale network. Simulation results show that the advance of performance is quite significant.

Contents

1	Introduction	1
1.1	Background	1
1.2	Motivation	5
1.3	Thesis Organization	9
1.4	Contribution	10
2	Overview of Self-Sizing Network	11
2.1	The Self-sizing Network Architecture	11
2.2	The Self-sizing Scheme	14
2.3	Traffic Measurement and Evaluation	16
2.4	The Fast Analysis and Synthesis Algorithm	17
3	Effective Bandwidth and its Measurement	19
3.1	Many Sources and Large Buffer Asymptotic (MSLBA)	19
3.2	Off-line Measurement of Effective Bandwidth	22
3.2.1	Measurement of $\alpha_j(s^*, t^*)$	23
3.2.2	The limitation of off-line measurement of EB	25
3.3	On-line Measurement of Effective Bandwidth	25
3.3.1	Accelerating the calculation of $\alpha_j(s, t)$	26
3.3.2	Preventing the numerical overflow	28
3.3.3	Effectiveness analysis for the DRDMW algorithm	30

4	Large-scale Optimal Band Partitioning Problem and its Solution	34
4.1	Optimal Band Partitioning (OBP) Problem	34
4.2	Simulated Annealing (SA) Algorithm	36
4.3	A SA-based Solution to OBP Problem	39
4.4	A Large-scale OBP Problem	44
4.4.1	Active SD node pairs	45
4.4.2	Quantity of links	46
4.5	A Hierarchical Self-Sizing Framework	50
5	Conclusions and Future Research	54
5.1	Conclusions	54
5.2	Future Research	56

List of Figures

1.1	The call acceptance control function with the self-sizing function . . .	6
2.1	The architecture of a self-sizing network	12
2.2	The deployment of rate regulators in the enhanced switching node .	13
2.3	The deployment of traffic monitors in the enhanced switching node	13
2.4	A link between two enhanced switching nodes	14
2.5	The self-sizing process	15
3.1	The illustration of the double sliding windows	28
3.2	The dual recursive double moving windows algorithm	31
3.3	The precision of the DRDMW algorithm	32
4.1	A practical self-sizing network	42
4.2	The average execution time vs the number of the active SD node pairs	46
4.3	A double-size self-sizing network	47
4.4	A quadruple-size self-sizing network	47
4.5	The average execution time of different network sizes	49
4.6	A hierarchical self-sizing framework	52

List of Tables

4.1	The bandwidth unit cost of simplex links in single-size network . . .	42
4.2	The workload of each traffic class b between each active SD node pair p	43
4.3	The average execution time vs the number of active SD node pairs .	46
4.4	The bandwidth unit cost of simplex links in quadruple-size network	48
4.5	The average execution time of different network sizes	49
4.6	The workload of each traffic class between each active SD node pair	51

Chapter 1

Introduction

1.1 Background

High-speed multi-service networks, capable of supporting voice, image, video and data traffic, have been strongly desired since these applications with diverse and stringent performance requirements soar within the existing networks. Originally, such a target might be achieved by realizing a B-ISDN network (Broadband Integrated Services Digital Network) based on ATM (Asynchronous Transfer Mode). The latter was indeed developed to combine the flexibility of packet switched solutions with the per-user quality of service (QoS) guarantees of traditional telephone networks. Nowadays, people expect to deploy a variety of services over a shared IP infrastructure due to its ubiquity.

Around 1993, comprehensive work began in the IETF to develop a mechanism that would allow IP to support more than a single best-effort class of service. This work resulted in the development of the Integrated Services (IntServ) whose architecture is based on *per-flow* resource reservation. However, IntServ is not scalable because it requires significant amounts of per-flow state and packet processing at each node along the end-to-end path but the resource reservation setup protocol (RSVP) does not have the ability to aggregate individually reserved sessions into a single shared class. Therefore, IntServ was not a suitable mechanism to support

the delivery of multiple classes of service in large IP networks[27].

By 1997, the IETF realized that IntServ was not going to be deployed in production networks and began to consider creating “relatively simple and coarse methods of providing multiple classes of service for Internet traffic.” As a result, the differentiated service (DiffServ) model was introduced, in which traffic that enters a network is first classified, marked and then possibly conditioned at the edges of the network according to its QoS requirements; then each packet is associated with one of behavior aggregates (BAs) via the Differentiated Services code-point (DSCP) carried in the packet’s header; when a packet enters the core of the network, each router along the transit path applies the appropriate per-hop behavior (PHB) based on the DSCP carried in the packet’s header[27].

Obviously, in the DiffServ networks, traffic flows¹ with various QoS requirements share network links, switching nodes, and processing facilities. Meeting the different service requirements is facilitated by sorting the traffic flows into their respective classes which would be treated differently, and the identical traffic class should be treated similarly across the network[27].

However, the DiffServ protocol performs admission control on a per-packet basis, not on a per-flow basis. Thus, the integration of aggregated RSVP at the edges of the network to perform per-flow admission control with DiffServ in the core of the network can much efficiently support QoS application on the Internet. Recently, researchers in the AQUILA and TEQUILA projects funded by European Union have already began to thoroughly investigate this solution to provide dynamic control to QoS based traffic[1][34].

As we know, both the IntServ and the DiffServ operates at Layer-3. If the routers in the core of the network run standard IP routing software (OSPF, IS-IS, and BGP-4) and operate conventional, longest-match IP forwarding, then all traffic flows would go through the shortest path. This may lead some links to be

¹A flow is a sequence of packets that have the same source and destination IP address and TCP or UDP port numbers

congested and other links unused. Therefore, we need traffic engineering to move traffic flows away from the more congested physical links and onto the potentially less congested ones across the network.

Multi-protocol label switching (MPLS), as a successful traffic engineering solution, can balance aggregate traffic load on the various links, routers, and switches in the network so that none of its individual components is over-utilized or under-utilized.

In the MPLS model, a label is a short, fixed-length value carried in the packet's header to map traffic to a specific Forwarding Equivalence Class (FEC). An FEC is a set of packets that are forwarded over the same Label Switching Path (LSP) through a network even if their ultimate destinations are different. An LSP is functionally equivalent to a virtual circuit because it defines an ingress-to-egress path, which is followed by all packets assigned to a specific FEC, through a network. The first label switch in an LSP is called the ingress, or head-end, label switch. The last label switch in an LSP is called the egress, or tail-end, label switch[26].

MPLS requires packet classification at the ingress edge of the network to assign an initial label to each packet. The ingress label switch then assigns a label to the packet and forwards it to the next hop in the LSP. In the core of the network, label switches swaps the incoming label with the outgoing label and directs the packet to the next hop in the LSP. When the labeled packet arrives at the egress label switch, the egress switch discards the label and forwards the packet using conventional longest-match IP forwarding[26].

Therefore, in order to support multi-service in large IP networks, MPLS may logically group flows of one traffic class between one pair of source-destination (SD) nodes into an FEC and associates the FEC with an LSP, which should be set up beforehand to reserve certain amount of resource according to the QoS requirements for the traffic class. Traffic flows within an FEC share buffers and link capacities reserved by the LSP.

Furthermore, we aggregate all LSPs corresponding to the identical traffic class

on each link into a “band” so that traffic flows of the same band can share the resources; hence each band which corresponds to a traffic class constructs a virtual subnetwork on the top of the physical network and different virtual subnetworks can meet varying QoS requirements for heterogeneous traffic. Thus, we physically partition network resources among bands.

Although traffic classification and network resource partitioning can meet different QoS requirements of multiple services, it also leads to low utilization of network capacity because only fully sharing of network resources can achieve high efficiency. Accordingly, A compromise has to be made between two conflicting demands: high network utilization and resource partitioning motivated by different QoS requirements while designing high-speed multi-service networks.

Since the traffic workload of each class varies frequently and dramatically, in order to achieve high efficiency within multi-service networks, we have to dynamically partition network resources among bands and frequently reconfigure each band to temporally and spatially track the varying traffic within the band under the limitation of the fixed overall link capacities. Hence elastic boundaries between successive bands are indispensable.

Actually, the elastic multi-service network design process involves two main steps: determining the overall network topology and capacities as well as partitioning the capacity among the network bands. The aggregate network design is a slow process based on a long-term forecast of the overall traffic, and is normally performed “off-line”. In contrast, network bands must be adjusted frequently (every few seconds) based on fresh traffic data and network-state information[36].

Although high-speed multi-service networks can achieve a comparable efficiency if the resource allocated to each band is adjusted to closely follow the traffic variations, up till now, even the most promising MPLS supporting DiffServ only relies on static provisioning and allocation of network capacities because how to dynamically allocate resources to each band in the MPLS architecture is still open area with many unresolved issues, e.g. how to describe, distribute, analyze and synthe-

size real-time traffic and state information of the network? In this thesis, we try to address this issue by introducing *self-sizing* mechanism to the MPLS architecture.

1.2 Motivation

Self-sizing was first proposed as a novel mechanism of resource management to handle dynamic resource allocation in ATM. At the very beginning, *self-sizing* means the autonomous adjustment of virtual path (VP) bandwidth based on traffic conditions observed in real time[25]. Then, it was expanded to an optimized adaptive resource allocation scheme that exploits one fast network analysis and synthesis algorithm to handle the real-time traffic information obtained via traffic evaluation and projection in the literature[36].

In the MPLS model, resource allocation is achieved by setting up LSPs. Thus, as an optimized adaptive resource allocation scheme, the *self-sizing* mechanism in the MPLS architecture focuses on whether LSPs can be set up and how LSPs are routed according to the available capacities.

It is the connection admission control (CAC) that determines whether an LSP can be set up or not. Therefore, we need to know the required capacities by the incoming traffic flows, the consumed capacities by the existing traffic flows and the total physical capacities.

We should combine the declaration based admission control (DBAC) with the measurement based admission control (MBAC) in the case of MPLS because, on the one hand, the incoming traffic load is difficult to know *a priori* and the peak rates of the incoming traffic flows may be declared to describe their workload; on the other hand, we still need to estimate the consumed link capacities by the existing traffic load so as to precisely report the left link capacities to the CAC algorithm and adaptively assign resource to the existing traffic, which is referred to as “self-sizing”.

Among some approaches estimating the consumed capacities by the existing

traffic flows, *effective bandwidth* (EB)[9][16] has been proved to be the most feasible because only *effective bandwidth* can take the QoS requirements of traffic into account[30].

Due to linear relation of effective bandwidth, we can apply the concept of *effective bandwidth* to the *self-sizing* scheme. Furthermore, the call acceptance control algorithm with the *self-sizing* function can be detailed via EB measurements on Figure 1.1.

Figure 1.1: The call acceptance control function with the self-sizing function

Figure 1.1 shows that the scheme of the CAC algorithm in one traffic class. Herein, we should know the workload of both incoming calls and active calls so as to decide whether the available bandwidth is sufficient to satisfy the QoS requirements of the incoming calls without degrading the service to already active calls. Therefore, for those incoming calls that have not yet available on the link, we have to claim some value to describe their workload beforehand.

If the entire finite realization of the incoming calls is known beforehand, as is the case for pre-recorded video, we can use the off-line EB measurement to estimate their workload; if their finite realization is unknown, we have to declare the peak rates (p_{k+1} on Figure 1.1) of the incoming calls to describe their workload. For those active calls that have been available on the link, then we can on-line estimate the EB of active calls so as to measure their workload.

When a new request arrives, the new call declares a peak rate or its off-line estimated EB if known beforehand. The information related to the new required link capacity is sent to the central controller via high priority connection and is summed to the on-line estimated consumed link capacity by the fast analysis and synthesis algorithm. If the total is less than the output link capacity C , then the new call is accepted. Then the link partition will be implemented through rate regulator. Thus, the self-sizing mechanism can not only *statistically* guarantee the required QoS to all connections but also improve the utilization of network resource via adaptive resource allocation.

The link partition determined by the fast analysis and synthesis algorithm acts conservatively. On the one hand, some of incoming calls claim the peak rates for the required link capacity; on the other hand, the actually link capacity consumed by the active calls decreases after the information of link capacity already in use is sent out and before the new link partition takes effect because the new calls do not yet come in and some active calls die out

As soon as new calls start transmitting data, the *self-sizing* mechanism begins to update the estimation of the total consumed link bandwidth and the rate regulators adjust the service rates of buffers to assign the appropriate link capacity to those active calls via the on-line EB measurement of active calls. Thus, the precise and prompt measurement of EB for estimating the workload of active calls is quite critical to the self-sizing mechanism.

In the literature [11][33], the authors compared four methods measuring effective bandwidth: a direct estimator, a block estimator, an estimator based on the Kullback-Leibler distance and an estimator based on linear regression. They also measured the EB values of the finite time realizations of On-Off fluids, Poisson arrival processes in the connection admission control (CAC) problem. Finally, they concluded that the resource provisioning based on the measured EB values could guarantee the required QoS in terms of cell loss rate (CLR).

However, their studies are based on the assumption that the entire trace must

have been available beforehand for its EB to be measured. Nonetheless, in the self-sizing scenario, the workload of the existing traffic of each class is unknown and quite volatile; hence how to on-line estimate the consumed capacities of the existing traffic via measuring their *effective bandwidth* should be answered. Otherwise, self-sizing mechanism would not make any sense. So we seek to propose an approach to implement on-line measurement of the effective bandwidth in this thesis.

After the consumed network capacities by the existing traffic load are known, a fast network analysis and synthesis algorithm should be introduced to dynamically determine whether LSPs can be set up or not. Since the total allocations must be within the aggregate network capacities, the capacity allocation of each band cannot be done independently of the allocations to other bands. Moreover, allocation updates must be completed within few seconds periodically. Therefore, all bands of the network should be sized simultaneously.

Furthermore, in addition to adaptive link partitioning, constrained-based LSP routing should be exploited to realize more flexible controls to balance the traffic intensity across the network and increase the utilization of network capacity. Thus, every LSP should have the alternative of different paths with distinctive usage cost. Consequently, while we allocate resources among bands to meet the different QoS requirements of traffic classes, how to route LSPs through the whole network to minimize the total network usage cost under the constraint of the overall physical capacities turns out to be a complex combinatorial optimization problem.

An optimization model has been developed to partition link capacity among bands to minimize network usage cost under the overall physical constraint of capacity and a fast algorithm simulating the *annealing process* has been also proposed to solve this combinatorial optimization problem in the literature[13].

Generally speaking, the execution time needed by the *simulated annealing* (SA) algorithm to find a global optima will rise explosively when the size of combinatorial optimization problem increases. Consequently, the execution time of the fast analysis and synthesis algorithm based on the *simulated annealing* technique will

likely violate the timeliness required by *self-sizing* in the very-large-scale network. How to overcome this violation should be figured out. Otherwise, *self-sizing* mechanism would also fail. So, in this thesis, we analyze the scalability of the *simulated annealing* algorithm in the *self-sizing* network and propose a hierarchical *self-sizing* framework and a distributed approach so as to shorten the time for the analysis and synthesis algorithm to obtain an optimal solution.

By and large, we have recognized that the timeliness of the *self-sizing* mechanism requires both the prompt report of traffic loads and the instant decision of link partitioning. As long as the interval for resource allocation is short compared with the time-scale of the traffic fluctuation, the network with the *self-sizing* capability should always be able to efficiently provide a satisfactory QoS level. Thus, we will tackle the timeliness of *self-sizing* from the perspective of the on-line EB measurement as well as the hierarchical *self-sizing* framework, respectively.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2: Provides an overview of both the logic function of the *self-sizing* mechanism and the network architecture implementing the above function, with emphasis on topics that relate to traffic evaluation and projection as well as the fast analysis and synthesis algorithm.

Chapter 3: Introduces the notion of the *effective bandwidth* and its approximation from the analysis of the *many sources and large buffer asymptotic*. Also reviews off-line measurement techniques of *effective bandwidth* in the scenario of self-sizing. And derives a dual recursive algorithm with double moving windows (DRDMW) used to on-line measure the EB of the traffic. Last discusses the effectiveness of the proposed DRDMW algorithm.

Chapter 4: Formulates a combinatorial optimization model to describe the op-

timal band partitioning (OBP) problem. Also provides a fast analysis and synthesis algorithm based on the *simulated annealing* technique to solve this OBP problem. Examines the scalability of the fast analysis and synthesis algorithm and presents a hierarchical *self-sizing* framework and a distributed approach so as to implement the *self-sizing* mechanism in the large-scale network.

Chapter 5: Draws concise conclusions and highlights outstanding issues for future work.

1.4 Contribution

The following are the main contributions of this thesis:

- We propose a dual recursive algorithm with double moving windows (DRDMW) to accelerate the calculation of EB value and prevent potential numerical overflow so as to implement on-line measurement of EB. Furthermore, we conduct some simulations to validate the effectiveness of the algorithm by focusing on the self-similar traffic ($1/2 < H < 1$) that implies long-range dependence[23].
- We find that the performance (the execution time) of the fast analysis and synthesis algorithm based on the *simulated annealing* technique deteriorates when the network scales up in terms of the number of both links and active SD node pairs. Then, we propose a hierarchical *self-sizing* framework and a distributed approach to split a large network into several small subnetworks so that we can simplify one complex combinatorial optimization problem into some simple combinatorial problems to make full use of parallel processing capacity of the network. Finally, the simulation results shows that the proposed approach can lessen the execution time significantly but only slightly deteriorate the optima.

Chapter 2

Overview of Self-Sizing Network

The network capable of *self-sizing* is able to sort traffic into classes, partition link capacities into bands corresponding to traffic classes, on-line measure traffic load of bands within links, synthesize real-time traffic and network state information, determine link capacity allocation among bands and adjust the size of bands or the service rate of traffic classes. Therefore, the *self-sizing* is both a network-wide service rate control mechanism and a dynamic resource allocation process.

This chapter mainly covers the novel approach of self-sizing. We first describe the network architecture of *self-sizing* through a series of illustrative figures. We then move on to the detailed scheme of the *self-sizing*. These foundations will be necessary to understand two challenges we may encounter to develop the *self-sizing* network: traffic evaluation and projection as well as the decision on the fast analysis and synthesis algorithm, which will also be covered in this chapter.

2.1 The Self-sizing Network Architecture

Since *self-sizing* is a network-wide control mechanism in the high-speed, multi-service network, a possible implementation is to employ a central controller as a server logically connected to one of the switching nodes. The controller interacts with the participating nodes through high-priority connections.

Figure 2.1: The architecture of a self-sizing network

Figure 2.1 illustrates an architecture of a *self-sizing* network, in which all switching nodes will be enhanced to include rate regulators associated with output links to implement link partitioning, and some traffic monitors related to each traffic class of each input link to perform traffic measurement. One of the enhanced switching nodes should be further updated to embody a central controller. Consequently, this node could logically work as a server to process fresh traffic and network state information via a fast analysis and synthesis algorithm, and then to make a decision to adjust allocation of link capacity among bands.

Because all of output links of the enhanced switching node may be partitioned differently, each output link needs one rate regulator to control all service rates of buffers associated with every traffic class through each output link. On the other hand, because each input link may contain all the possible bands and traffic in each band must be monitored, traffic monitors should be deployed for each band of every input link.

Although an enhanced switching node contains both rate regulators and traffic monitors, we still illustrate separately their deployments to present them clearly. Figure 2.2 shows possible deployment of rate regulators in one enhanced switching node. Figure 2.3 presents reasonable installment of traffic monitors. Actually,

Figure 2.2: The deployment of rate regulators in the enhanced switching node

Figure 2.3: The deployment of traffic monitors in the enhanced switching node

Figure 2.4: A link between two enhanced switching nodes

Figure 2.2 and Figure 2.3 should merge; hence each duplex link is associated with one rate regulator and a group of traffic monitors at each end. Figure 2.4 illustrates how two adjacent enhanced switching nodes partition a link.

2.2 The Self-sizing Scheme

As a dynamic resource allocation process in the high-speed, multi-service network, the *self-sizing* follows the following steps, as illustrated in Figure 2.5, to implement network-wide service rate control mechanism:

1. The overall physical capacity of links are configured in the central controller.
2. The individual enhanced switching nodes periodically report the real-time network state information (possible link failures and/or recoveries) to the central controller in the server.
3. Traffic monitors embedded in the enhanced switching nodes measure the real-time traffic loads of the individual classes and report traffic intensities periodically to the central controller in the server.

Figure 2.5: The self-sizing process

4. The central controller freezes previously accepted allocations of k bands, allots the remaining link capacity to the unserved demand among bands with iterative adjustments and identifies possible link-capacity violations.
5. The central controller terminates the successive allocation process when no further progress is possible and the unserved demand, if any, is rejected.
6. The central controller delivers the commands involving link partitioning to the switching nodes
7. The rate regulators in the switching nodes adjust the service rates of buffers associated with each traffic class to partition the output links.

Since traffic load of each individual class is more volatile than the total traffic that is, of course, a moving target, the *self-sizing* mechanism requires the capability

of real-time reporting of traffic and network state information as well as a fast network analysis and synthesis algorithm to make quick decision on link partitioning.

2.3 Traffic Measurement and Evaluation

Traffic measurement and evaluation to promptly report real-time traffic load is the first challenge that we meet in the *self-sizing* network. Due to the **delays** in a fraction of one second incurred by transferring measurements from the traffic monitors in the enhanced switching nodes to the controller in the server, the **execution time** on the order of seconds needed by the network analysis and synthesis algorithm at the controller, the **delays** with millisecond induced by transferring commands from the controller to the rate regulators, the rate regulator can only take effect afterwards. Therefore, not only do we promptly evaluate the amount of bandwidth needed for each traffic pair to satisfy the required QoS at the cell level but also, more importantly, predict the likely traffic condition when the rate regulator functions.

Since the packet arrival process of each traffic class in the high-speed multi-service network is a superposition of sources which are typically bursty and highly correlated between different times, the *self-sizing* mechanism requires a traffic descriptor able to characterize each traffic connection requiring a probabilistic guarantee of QoS. Although the peak rate of the traffic can provide a deterministic guarantee of QoS and meet future demands, it leads to over-provisioning and inefficiency. The average rate of the traffic might be employed to improve the efficiency but the QoS requirements could be violated. The *effective bandwidth* of traffic should be an intuitive choice because it is related with the QoS performance and takes the complete profile including future trend of the traffic into account through *large deviation theory*[16].

In Chapter 3, we will introduce the notion of *effective bandwidth*, *effective bandwidth function* and *effective bandwidth approximation* in terms of *the many sources*

asymptotic. Also, we will discuss both off-line and on-line approaches measuring the *effective bandwidth*.

2.4 The Fast Analysis and Synthesis Algorithm

The analysis and synthesis algorithm must first take the constraint of the aggregate network capacity into account when it determines how to allocate network capacity among multiple bands; hence the capacity allocation to one network band cannot be done independently of the allocations to other bands

The *sequential* approach, a simple technique for multi-band sizing, is to size the bands sequentially while observing the total capacity limitation. That is to say, a given band is sized to serve its traffic at the required QoS within the limitation of the total network capacity. The next band is then sized under the constraint of the remaining capacity. The available capacity is updated after each band is considered. A cyclic fashion to choose the starting band is employed to prevent potential unfairness when allocation adjustments proceed periodically.

However, this *sequential* approach is slow and infeasible with a large number of nodes because it is quite necessary for a network analysis and synthesis algorithm, capable of sizing networks of several hundred nodes, to have the execution time on the order of several seconds[36]. Thus, it is the second challenge for us to find a fast analysis and synthesis algorithm on a large-scale network due to the timeliness of the *self-sizing* mechanism.

Fortunately, the *parallel* approach, an alternative technique which can benefit from parallel processing, is able to size all the bands simultaneously with iterative adjustments when the constraint of link capacity is violated. Through this approach, a dynamic selective routing, in which the controller determines either a direct route or an alternative one for each source-destination (SD) pair, can be employed to balance the traffic intensities concurrently. Algorithms for combining parallel partitioning with dynamic routing have been proposed for connection-based

networks and other multi-service networks, as described in [37].

Even if the *parallel* approach has been proved effective and feasible, we are still concerned about its scalability on the large-scale network because the *self-sizing* is a network-wide control mechanism. In Chapter 4, we will present a mathematical model for band partitioning related to the *parallel* approach, formulate a combinatorial optimization problem and review a search technique — simulated annealing algorithm — to solve this problem. Also, we investigate the scalability of the simulated annealing algorithm in the scenario of *self-sizing*.

Chapter 3

Effective Bandwidth and its Measurement

3.1 Many Sources and Large Buffer Asymptotic (MSLBA)

Roughly speaking, *Large Deviation* is a theory of rare events. One of its applications is to investigate the tails of probability distributions and, in recent years, this aspect of the theory has been widely used in queuing theory to analyze how the buffer overflow probability decays asymptotically as some quantity increases. If the buffer size per source and link capacity per source are kept constant, and the number of the multiplexed sources grows, then the buffer size and the link capacities enlarge in the meanwhile. Thus, we have the *many sources and large buffer asymptotic* (MSLBA) [3][5][31].

Assume the arrival process at a broadband link is the superposition of independent sources of J types. Let n_j be the number of sources of type j , and let $n = (n_1, \dots, n_J)$. The broadband link has a shared buffer of size $B = Nb$ and link capacity $C = Nc$. Parameter N is the scaling parameter and parameters b, c are the buffer size and capacity per n sources, respectively.

Let $X_j[0, t]$ be the total load produced by a source of type j in the time interval $[0, t]$, which feeds the above link. Herein, we assume that $X_j[0, t]$ has stationary increments. And also let Q be the queue length in an infinite buffer which multiplexes $N * n = (N * n_1, \dots, N * n_J)$ sources and is served at rate $C = N * c$. Then, we denote with $P(Q > Nb)$ the probability that the queue length Q is above the threshold $B = N * b$. Courcoubetis *et al.* in [5] established

$$\lim_{N \rightarrow \infty} \frac{1}{N} \log P(Q > Nb) = \sup_t \inf_s \left[st \sum_{j=1}^J n_j \alpha_j(s, t) - s(b + ct) \right] = -I \quad (3.1)$$

where I is called the *asymptotic rate function*, and

$$\alpha_j(s, t) = \frac{1}{st} \log E \left[e^{sX_j[0, t]} \right], \quad (3.2)$$

where s, t are system parameters that characterize the *context* of the source, which includes the link resources (capacity and buffer), scheduling discipline, the QoS requirements, and traffic mix (percentage and characteristics of different source types).

Specifically, the *time* parameter t corresponds to the most probable duration of the buffer busy period prior to overflow. For the small buffer size and the fixed capacity, the buffer overflow might occur frequently due to the fast traffic burstiness, hence the small buffer size implies a small value of t ; on the contrary, when the buffer size becomes large, the buffer overflow turns to be less possible and the duration of non-overflow might increase. Furthermore, parameter t shows the minimum time granularity that the arrival processes need to capture the statistical properties of the traffic that affect buffer overflow.

A similar asymptotic analysis holds for the proportion of workload lost through the overflow of a finite buffer size $B = Nb$. Due to expression (3.1), the overflow probability can be written as

$$P(\text{overflow}) \approx e^{-NI} \quad (3.3)$$

when N is large. Consider the QoS constraint on the overflow probability to be

$P(\text{overflow}) \leq e^{-\gamma}$ and combine this with (3.1), then we have, for large N ,

$$\sum_{j=1}^J N * n_j \alpha_j(s, t) \leq C + \frac{1}{t} (B - \frac{\gamma}{s}). \quad (3.4)$$

This suggests that *effective bandwidth* through MSLBA can be employed to obtain a circuit-switched style call acceptance control (CAC) algorithm for packet-switched networks.

And then take the derivative over (3.4)(see also [4]), we have

$$s = \frac{\partial \gamma}{\partial B} \quad \text{and} \quad st = \frac{\partial \gamma}{\partial C}. \quad (3.5)$$

Therefore, the *space* parameter s (measured in e.g., kb^{-1}) is equal to the rate at which the logarithm of the overflow probability decreases with the buffer size for the fixed capacity. For the small buffer size one would expect s to be large because the buffer is used to smooth the fast traffic burstiness and reduce the overflow probability dramatically. Once the fast traffic burstiness has been smoothed, the slow traffic burstiness governs buffer overflow. Thus, when the buffer size is large, increasing the buffer has a very small effect on the overflow probability and s is small. In addition, the product st indicates how much the logarithm of the overflow probability decreases with the link capacity for the fixed buffer size.

Expression (3.1) is referred to as the *many sources and large buffer asymptotic*, and $\alpha_j(s, t)$ is defined as the *effective bandwidth function* of a source of type j through (3.2)[6][16]. Corresponding to one specified *context*, namely *a particular operating point* of the link, expressed through the parameter pair (s^*, t^*) , $\alpha_j(s^*, t^*)$ is known as the *effective bandwidth* of a source of type j [20].

Thus, the *effective bandwidth* $\alpha_j(s^*, t^*)$ based on the *many sources and large buffer asymptotic* can characterize resource usage which depends not only on the statistical properties and the QoS requirements of a source, but also on the statistical properties of the other traffic it is multiplexed with and the resources (capacity and buffer) of the multiplexing link.

The effective bandwidth function shown in (3.2) requires arrival rate process knowledge and is difficult to compute. Nevertheless, the effective bandwidth func-

tions for some particular arrival rate processes (e.g. ON-OFF, Poisson, FBM, MMPP models) might be further detailed based on the specific parameters of the traffic model. For example, a Gaussian source

$$X[0, t] = \lambda t + Z(t), \quad (3.6)$$

where λ is the mean rate of the arrival rate process and $Z(t)$ is normally distributed with zero mean, has the *effective bandwidth function* as follows:

$$\alpha(s, t) = \lambda + \frac{s}{2t} \text{Var } Z(t). \quad (3.7)$$

Despite the elegant and compact formula of the *effective bandwidth function* like the above expression, it is non-trivial to determine a traffic model as well as its parameters of an arrival rate process in the complex scenario of modern broadband networks beforehand. Consequently, the most common EB-based techniques utilize traffic measurements to estimate the EB values associated with traffic flows.

Next, we first review the off-line measurement of EB of the known traffic, which has been discussed in the literature[13], where the authors prove the EB's effectiveness in the scenario of *self-sizing*. Then we move on to the on-line measurement of the effective bandwidth which is still an open field.

3.2 Off-line Measurement of Effective Bandwidth

Measuring the *effective bandwidth* of a source of type j is exactly to evaluate one specific value of the *effective bandwidth function* of the source of type j in a particular *context*, in which all relevant factors noted above have been determined and can be characterized through the parameters s^*, t^* .

Generally speaking, we should employ the *effective bandwidth function* (3.2) under the *many sources and large buffer asymptotic* since this definition is the most complete and can be applied to any contexts and traffic models. Thus, in order to obtain the value of $\alpha_j(s^*, t^*)$, one has to, on the one hand, approximate the expectation $E[e^{sX_j[0, t]}]$ in *effective bandwidth function* (3.2) of a source of type j through

measuring real traffic trace and, on the other hand, find a *particular operating point* (s^*, t^*) corresponding to that specific *context* in which the measurement is taking place.

3.2.1 Measurement of $\alpha_j(s^*, t^*)$

$E[e^{sX_j[0,t]}]$ **approximation:**

Given a sample trace of type j , in which each packet is assumed to have a corresponding record giving its arrival time and size. We shall represent this information by the collection $\chi_j = \{(t_n, x_n); n = 1, \dots, N_j\}$ for a sample trace consisting of N_j packets with arrival time and size given by t_n and x_n , respectively. Then, we define $X_j[\tau, \tau + t]$ as the amount of data arriving during the interval $[\tau, \tau + t]$. Thus, we could have

$$X_j[\tau, \tau + t] = \sum_{n=1}^{N_j} x_n I(\tau \leq t_n \leq \tau + t) \quad 0 \leq \tau \leq t_{N_j} - t, \quad (3.8)$$

where

$$I(\tau \leq t_n \leq \tau + t) = \begin{cases} 1 & t_n \in [\tau, \tau + t] \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

Assume that $X_j[0, t]$ is an arrival process with a stationary and ergodic increment. Gibbens proposed to approximate the probabilistic expectation $E[e^{sX_j[0,t]}]$ with the temporal average of the underlying process and (3.2) can be rewritten as follows[12]:

$$\alpha_j(s, t) = \frac{1}{st} \log \frac{1}{t_{N_j} - t} \int_0^{t_{N_j} - t} e^{sX_j[\tau, \tau + t]} d\tau. \quad (3.10)$$

Combining (3.8) with (3.10), we could have

$$\alpha_j(s, t) = \frac{1}{st} \log \frac{1}{t_{N_j} - t} \int_0^{t_{N_j} - t} e^{s \sum_{n=1}^{N_j} x_n I(\tau \leq t_n \leq \tau + t)} d\tau, \quad (3.11)$$

where the integrand $e^{s \sum_{n=1}^{N_j} x_n I(\tau \leq t_n \leq \tau + t)}$ is a piece-wise constant function with jumps, which means that its function value only needs to be recalculated, at the time epochs when either a packet enters or leaves the recording interval $[\tau, \tau + t]$ while using a sliding window of width t .

Since the above (3.11) originates from the definition of the *effective bandwidth function* based on the *many sources and large buffer asymptotic*, it is not restricted by any additional assumption and can be applied to either long range dependent traffic sources or short range dependent ones. However, a large number of arrivals of the underlying process within $[\tau, \tau + t)$ might lead to the computation overflow when we calculate the integral in (3.11). Thus, the numerical precision of CPU may limit the applicability of (3.11) in both parameter spaces.

If the arrivals of the underlying process within non-overlapping blocks $[\tau, \tau + t)$ are realizations of independent and identically distributed (i.i.d.) random variables, a simplified approximation of $E[e^{sX_j[0,t]}]$, which has been proposed in [7], can be employed to take non-overlapping blocks into account. It takes the form as follows:

$$\alpha_j(s, t) = \frac{1}{st} \log \frac{1}{\lfloor \frac{tN_j}{t} \rfloor} \sum_{k=1}^{\lfloor \frac{tN_j}{t} \rfloor} e^{sX_j[(k-1)t, kt]}. \quad (3.12)$$

Although (3.12) is also derived from the definition of the *effective bandwidth function* based on the *many sources and large buffer asymptotic*, the *i.i.d.* assumption makes this estimator only applicable to short range dependent traffic source. Duffield *et al.* demonstrate that this approximation is asymptotically unbiased for the short range dependent traffic as the number of blocks in the sample trace approaches to infinity[7].

Search for (s^*, t^*) :

In the scenario of self-sizing, all active calls of one traffic class share one buffer and can still be modeled by the *many sources and large buffer asymptotic*. Moreover, we assume that the values of effective bandwidth of all flows in one traffic class are equal.

Given the shared buffer size B , the maximum buffer overflow probability Γ , the number n_j of active calls as well as one of per-flow traffic traces, according to the expression (3.1), the consumed capacity C^* by the active calls can be computed by

solving the equation as follows[32]:

$$C^* = \sup_t \inf_s R(s, t), \quad (3.13)$$

where

$$R(s, t) = n_j \alpha_j(s, t) - \frac{\log \Gamma}{st} - \frac{B}{t}. \quad (3.14)$$

To solve the equation (3.13), we fix t to find the minimum $R^*(t) = \min_s R(s, t)$ and $s^* = \arg \min_s R(s, t)$, then find the maximum $C^* = \max_t R^*(t)$ and $t^* = \arg \max_t R^*(t)$. Details can be found in [5].

3.2.2 The limitation of off-line measurement of EB

For the estimation of $\alpha_j(s^*, t^*)$, we need the whole finite trace of the traffic beforehand to search for (s^*, t^*) using the **supinf** formula (3.13); furthermore, the run-time required for numerically solving the **supinf** formula (3.13) depends primarily on the size (number of epochs) of the trace file and the range of values of t that are linearly searched. For example, when the trace files contain 40,000 epochs and 50 values of t are searched, the solution of the **supinf** formula requires approximately 23 seconds on an ULTRA-1 workstation with one UltraSPARC processor at 170 MHz.[5]

As a summary, the estimation of $\alpha_j(s^*, t^*)$ does not only require that the finite realization of the traffic is available beforehand but also takes too long time to meet the timeliness of the real-time applications. Thus, the above approaches of EB estimation are only applicable to off-line estimation of EB, where the finite trace is known beforehand.

3.3 On-line Measurement of Effective Bandwidth

For the traffic whose not only finite realization is unknown in advance but also EB should be on-line reported, like the active calls in the scenario of *self-sizing*,

we need to develop a novel approach to on-line measure the EB of these real-time traffic.

Based on the above discussion concerning the off-line measurement of EB, we have noticed that we might on-line measure the EB of those real-time traffic available on the link as long as we manage to accelerate the off-line measurement of EB[29]. However, the speed bottleneck of the off-line EB measurement lies in the calculation of $\alpha_j(s, t)$ in the equation (3.14).

3.3.1 Accelerating the calculation of $\alpha_j(s, t)$

Since (3.11) holds for any type j of traffic, it could be simplified and re-presented as follows:

$$\alpha(s, t) = \frac{1}{st} \log \frac{1}{t_N - t} \int_0^{t_N - t} e^{s \sum_{n=1}^N x_n I(\tau \leq t_n \leq \tau + t)} d\tau, \quad (3.15)$$

Given s^* and t^* , the integral in (3.15) can be readily computed without truncation error because the integrand $e^{s \sum_{i=1}^N x_i I(\tau \leq t_i \leq \tau + t)}$ is a piece-wise constant function. Assume that the average arrival rate of traffic is M (pkts/sec) and the length of the available trace measured until now from the onset of real traffic is t_N (sec), we need to calculate $e^{s \sum_{i=1}^N x_i I(\tau \leq t_i \leq \tau + t)}$, multiplications and additions, respectively, as many as $2(t_N - t)M$ times.

When the average arrival rate M is quite high, the calculation workload for estimating $\alpha(s^*, t^*)$ might be extremely heavy. Consequently, it is possible for the computation to cost much more time than the duration t_N of the available trace. Therefore, we must manage to reduce the computation complexity of the integral in (3.15) so as to accelerate the calculation of the integral.

If we calculate $e^{s \sum_{i=1}^N x_i I(\tau \leq t_i \leq \tau + t)}$ in (3.15) only once when m packets in average either enter or leave the recorded interval $[\tau, \tau + t]$, then the above integrand will be calculated once over the interval of $\frac{m}{M}$ (sec), which we define as ONE *epoch*. In addition, we can count the number of packets arriving and leaving the recorded interval $[\tau, \tau + t]$ concurrently at the end of each epoch. Thus, the computation

complexity of the integral in (3.15) is reduced $2m$ times. It means that we need to calculate $e^{s \sum_{i=1}^N x_i I(\tau \leq t_i \leq \tau+t)}$, multiplications and additions, respectively, as many as $(t_N - t) \frac{M}{m}$ times.

Then, one trace that is $t_N = \frac{Jm}{M}$ long can be represented as the collection $Z = \{(j, z_j), j = 1, \dots, J\}$, where z_j is referred to as the aggregated workload arriving within the j -th epoch, namely,

$$z_j = \sum_{i=1}^N x_i I[(j-1) \frac{m}{M} \leq t_i \leq j \frac{m}{M}] \quad j = 1, \dots, J. \quad (3.16)$$

Given the time scale $t = \frac{Km}{M}$, then we rewrite (3.15) as follows:

$$\alpha(s, K, J, m, M) = \frac{M}{sKm} \log \frac{1}{(J-K)} \sum_{l=1}^{J-K+1} e^{s \sum_{i=1}^N x_i I[(l-1) \frac{m}{M} \leq t_i \leq (l-1+K) \frac{m}{M}]}. \quad (3.17)$$

Provided

$$z(j_1, j_2) = \sum_{i=1}^N x_i I[(j_1-1) \frac{m}{M} \leq t_i \leq j_2 \frac{m}{M}] \quad j_1, j_2 = 1, \dots, J. \quad (3.18)$$

Then, from (3.18), we re-state (3.17) as follows:

$$\alpha(s, K, J, m, M) = \frac{M}{sKm} \log \frac{1}{(J-K)} \sum_{l=1}^{J-K+1} e^{sz(l, l+K-1)}. \quad (3.19)$$

Thus, through the introduction of an epoch, we are able to measure the EB of the available trace at the end of the J -th epoch, $2m$ times faster but somewhat less precise than before. At this point, we only need to calculate $e^{sz(l, l+K-1)}$ and additions as many as $(J-K)$ times, respectively, so as to obtain the value of the summation in (3.19).

Furthermore, we find that the measured value $\alpha(J+1)$ of the EB at the end of the $(J+1)$ -th epoch can be derived from those data collected within the $(J+1)$ -th epoch and the measured value $\alpha(J)$ of the EB at the end of the J -th epoch. The recursive relationship is expressed as follows:

$$\alpha(J) = \frac{M}{sKm} \log \frac{1}{(J-K)} \sum_{l=1}^{J-K+1} e^{sz(l, l+K-1)}, \quad (3.20)$$

$$\alpha(J+1) = \frac{M}{sKm} \log \frac{1}{(J-K+1)} \sum_{l=1}^{J-K+2} e^{sz(l, l+K-1)}, \quad (3.21)$$

and

$$\sum_{l=1}^{J-K+2} e^{sz(l, l+K-1)} = \sum_{l=1}^{J-K+1} e^{sz(l, l+K-1)} + e^{sz(J-K+2, J+1)}. \quad (3.22)$$

So far, we only need the calculation of $e^{sz(J-K+2, J+1)}$ and addition, respectively, as many as once so as to obtain the summation in (3.21). Thus the run-time to estimate $\alpha(s, t)$ ultimately depends on the processing capacity of computer system.

3.3.2 Preventing the numerical overflow

However, the numerical overflow might eventually occur with the trace length J approaching to infinity while we calculate the summation in (3.19). Thus, we instead choose one fixed length (J epochs) trace segment immediately before as a sample to calculate the freshest EB value of the real traffic. And we have to update this J epochs long sample trace when new packets arrive or old ones leave the sample trace immediately before as illustrated in Figure 3.1. Likewise, we estimate the most updated EB value once every *epoch*. Of course, we also sacrifice some precision.

Figure 3.1: The illustration of the double sliding windows

Inspired from (3.19), we can obtain the measured value $\alpha(n+1)$ of the EB at the end of the $(n+1)$ -th *epoch* based on those data collected within the $(n+1)$ -th

epoch and the measured value $\alpha(n)$ of the EB at the end of the n -th *epoch* ($n \geq J$) shown on Figure 3.1. The recursive relationship is expressed as follows:

$$\alpha(n) = \frac{M}{sKm} \log \frac{1}{(J-K)} \sum_{l=1}^{J-K+1} e^{sz(l+n-J, l+n+K-J-1)}, \quad (3.23)$$

$$\alpha(n+1) = \frac{M}{sKm} \log \frac{1}{(J-K)} \sum_{l=1}^{J-K+1} e^{sz(l+n+1-J, l+n+K-J)}, \quad (3.24)$$

and

$$\begin{aligned} \sum_{l=1}^{J-K+1} e^{sz(l+n+1-J, l+n+K-J)} &= \sum_{l=1}^{J-K+1} e^{sz(l+n-J, l+n+K-J-1)} \\ &\quad - e^{sz(n+1-J, n+K-J)} + e^{sz(n+2-K, n+1)}. \end{aligned} \quad (3.25)$$

Similarly, we only need the calculation of $e^{sz(n+1-K, n)}$, addition and subtraction, respectively, as many as once so as to obtain the summation in (3.24). Thus the run-time to estimate $\alpha(s, t)$ ultimately depends on the time-scale K and the processing capacity of computer system.

The above recursive relationship implies that we need to slide double windows, of which the larger and outer one is a J (*epoch*) wide sample trace window (e.g. $J = 1024$ *epochs*) and the smaller and inner one is a K (*epoch*) wide time-scale window (e.g. $K = 1$ *epoch*) shown on Figure 3.1. Every time we finish the collection of workload of the arrival process during the $(n+1)$ -th epoch, the outer window moves forward to contain the most recent $(n+1)$ -th epoch and discard the most obsolete $(n-J+1)$ -th epoch. After that, the inner time-scale window slides from the $(n-J+2)$ -th epoch to the $(n-K+1)$ -th epoch within the outer window.

Due to the potential numerical overflow of $\sum_{l=1}^{J-K+1} e^{sz(l, l+K-1)}$, the width of the sample trace window should be limited a small value. Thus, it is impossible for the measured value of the EB to characterize the whole traffic trace, especially under the assumption of long-range dependence (LRD).

Consequently, it is necessary for us to develop a second level recursive algorithm to estimate the characteristic value of the EB based on the measured value of the EB so as to take the whole traffic trace into consideration. Intuitively, we consider

to average all of the measure values of the EB. Thus, for $n \geq J$, we have

$$\bar{\alpha}(n) = \frac{\sum_{i=J}^n \alpha(i)}{n - J + 1} \quad (3.26)$$

and

$$\bar{\alpha}(n+1) = \frac{(n - J + 1)\bar{\alpha}(n)}{n - J + 2} + \frac{\alpha(n+1)}{n - J + 2}. \quad (3.27)$$

Finally, we formulate this dual recursive algorithm with double moving windows (DRDMW) to on-line estimate the EB by following (3.23), (3.24), (3.25), (3.26) and (3.27) step-by-step. From (3.26), we can see that the first characteristic value $\bar{\alpha}(n)$ of the EB is actually the first measured value $\alpha(n)$ of the EB when n equals J .

The dual recursive double moving windows (DRDMW) algorithm can be summarized as the following pseudo codes:

For given s and $t = \frac{Km}{M}$ as well as some traffic flow, the measured value of $\alpha_j(s, t)$ via this dual recursive algorithm with double moving windows (DRDMW) only depends on the window width J of sample trace. Thus, we will compare the precision of the proposed DRDMW algorithm with the original approach and take a look at how the window width J of sample trace affects its precision.

3.3.3 Effectiveness analysis for the DRDMW algorithm

In order to validate the effectiveness of the DRDMW algorithm, we need to choose a real-time traffic flow. Here, we prefer the long-range dependent traffic because it is much more bursty and representative than those traditional traffic. For the convenience of estimating the EB of self-similar traffic, we directly adopt a fast Fourier transform method proposed by Vern Paxson[24] to synthesize approximate self-similar flows for one type of self-similar process — fractional Gaussian noise (FGN) that is also a widely-studied fractal rate process.

To cite some intermediate results in the literature[11], the FGN trace has also been parameterized to match the Belcore rates. We still consider that the packets

```

010    initialize  $s$ ,  $K$  and  $J$ ;
020     $n := J$ ;
030    repeat
040        begin
050             $l := 0$ ;
060            repeat
070                begin
080                    calculate  $z(l + n - J, l + n - J + K - 1)$ ;
090                    calculate  $e^{sz(l+n-J, l+n-J+K-1)}$ ;
100                    slide the inner time-scale window  $l = l + 1$ ;
110                end
120            until  $l$  equals  $J - K + 1$ 
130            calculate  $\alpha(n)$ ;
140            calculate  $\bar{\alpha}(n)$ ;
150            slide the outer sample trace window  $n := n + 1$ ;
160        end
170    until the real traffic ends

```

Figure 3.2: The dual recursive double moving windows algorithm

are all of a fixed size. For example, $x_i = 53(\text{byte})$ would correspond to the size of an ATM cell, which often arises in practice.

Given the mean arrival rate $M = 14740 (\text{cell}/s)$ or $6.25 (\text{Mbps})$ and the variance $\sigma_M^2 = 4901796 (\text{cell}/s)^2$ of the traffic stream with a Hurst parameter of $H = 0.7$, we model the arrivals of cells in a discrete epoch with the average number of arrivals and the corresponding variance set to $m = 4 (\text{cell}/\text{epoch})$ and $\sigma_m^2 = 0.36 (\text{cell}/\text{slot})^2$, respectively. Hence, the length of an epoch is $0.271 (\text{ms})$. With these values, where not differently specified, the number of epochs in the stream was set to $L=262,144$,

corresponding to 1.18 minutes. Eventually, the FGN traffic stream is configured as the following:

$$\text{Mean: } m = 4 \text{ (cell/epoch)} \quad \text{Variance: } \sigma_m^2 = 0.36 \text{ (cell/epoch)}^2$$

$$\text{Epoch: } \mu = 0.000271 \text{ (s)} \quad \text{Length of stream: } L = 262,144 \text{ (epoch)}$$

Assume that the space-scale $s = 0.01 \text{ (1/cell)}$ and time-scale $K = 20 \text{ (epoch)}$.

First, we measure the EB according to the original definition in expression (3.11), then we employ the DRDMW algorithm to measure the EB under two different window width $J = 32$ and 1024 (epoch) of sample trace.

Figure 3.3: The precision of the DRDMW algorithm

From Figure 3.3, we can see that the measured EB values only increase less than 0.5 percent when the window width of sample trace extends 32 times. This result coincides with conclusions in [11]. Obviously, the measured EB values are quite close to the results via the original definition in (3.11) regardless of the window width J of the sample trace when the measurement process stabilizes. This also implies that J has little impact on the accuracy of the DRDMW algorithm.

However, we notice that the measured EB values via the DRDMW algorithm follow ones through the original definition more quickly when the window width J of the sample trace is smaller. More importantly, we can obtain the first measured EB

value more quickly when the window width J of sample trace is smaller. Moreover, the smaller values of J also makes potential numerical overflow less possible while we calculate the value of $\alpha(s, t)$. Thus, we prefer the smaller window width J of sample trace.

Chapter 4

Large-scale Optimal Band Partitioning Problem and its Solution

4.1 Optimal Band Partitioning (OBP) Problem

Consider a high-speed multi-service network that can provide three services: voice, video and data. The network can be divided into three virtual elastic bands: band 1 for voice, band 2 for video and band 3 for data. Each virtual band provides a single service with homogeneous QoS requirements.

In order to implement dynamic selective routing to balance traffic intensities, every traffic class for each source-destination pair is assigned two alternative routes. Generally speaking, the one is the path with the minimum costs and the other is with the second minimum costs when port processing cost and bandwidth usage cost are taken into account. Although dynamic routing can improve the utilization of network resource, the total system cost might increase dramatically because the path with the second minimum costs may be chosen instead of the one with the first minimum costs.

An optimization model to minimize the total system cost was proposed to size

the bands simultaneously in the literature[13]. The following notations are introduced to formulate the optimal band partitioning (OBP) problem.

J : set of simplex physical links. One duplex link includes two reverse simplex links.

CAP_j : total available capacity in basic bandwidth unit for link j . Here, we take 64kbps as the *basic bandwidth unit* (BBU).

$B = \{1, 2, 3\}$ set of bands, i.e. set of traffic classes.

w_{bj} : basic bandwidth unit cost for band b on link j , consisting of port processing cost and bandwidth usage cost.

E_b : effective bandwidth of one connection in traffic class b in basic bandwidth unit, derived from on-line measurement which guarantees cell loss probability. This quantity E_b is also known as the *service unit* of traffic class b . And each connection requires at least the same amount of link capacity as one service unit to meet its QoS requirement.

P : set of active source-destination (SD) node pairs.

v_{pb} : number of all connections in traffic class b for SD node pair p .

R_{pb} : set of two alternative routes for traffic class b for node pair p .

$A_{pbrj} = 1$ if the route $r \in R_{pb}$ for traffic class b for node pair p goes through link j , and 0 otherwise.

Then, we define the following decision variables:

C_{bj} : assigned capacity for band b in link j .

$x_{pbr} = 1$ if the traffic of class b at SD pair p goes through route $r \in R_{pb}$, and 0 otherwise.

The optimal band partitioning problem (OBP) can be formulated as finding C_{bj} and x_{pbr} values to minimize the total network cost, namely,

$$\min \sum_{b \in B} \sum_{j \in J} C_{bj} \cdot w_{bj} \quad (4.1)$$

s.t.

$$C_{bj} \geq \sum_{p \in P} \sum_{r \in R_{pb}} v_{pb} \cdot E_b \cdot A_{pbrj} \cdot x_{pbr} \quad b \in B, j \in J \quad (4.2)$$

$$\sum_{b \in B} C_{bj} \leq CAP_j \quad j \in J \quad (4.3)$$

$$\sum_{r \in R_{pb}} x_{pbr} = 1 \quad b \in B, p \in P \quad (4.4)$$

$$x_{pbr} = 0 \text{ or } 1, C_{bj} \geq 0 \quad p \in P, b \in B, j \in J, r \in R_{pb}. \quad (4.5)$$

Constraint (4.2) means that each band size partitioned to corresponding traffic class within each link should be large enough to meet the total capacity demand of all SD pairs of this traffic class traversing this link. The requirement for QoS is encoded in the effective bandwidth E_{pb} and will be guaranteed through the constraint in (4.2). Constraint in (4.3) shows that band partition of each link never violates the physical limit of link capacity.

OBP problem might be a large combinatorial optimization problem when a high-speed multi-service network has hundreds of nodes and thousands of links. It is difficult to find an optimization algorithm that can obtain an optimal result with reasonable computation effort[21], specially in the scenario of fast and frequent self-sizing. To solve this OBP problem, the authors in the literature[13] propose an algorithm based on *simulated annealing* (SA) and prove its effectiveness via simulation.

4.2 Simulated Annealing (SA) Algorithm

Simulated Annealing is a meta-heuristic based on the physical annealing process, through which the material is heated up to a very high temperature until it melts, and then is cooled down slowly into its ground state with minimum energy. While cooling, the material might try out its state with some energetic value. While it's still hot, the annealing process will experiment with quite different states (since the material is hot, its molecules can move around easily and try out states), but when it gets colder, the material will try to settle in those new states that have lower energetic values, and these states will not change that much since the lower temperatures keep the molecules from moving around.

Likewise, as a search technique, the simulated annealing algorithm starts out with some state that is often a solution for the combinatorial problem, and randomly visits neighbors of that solution. At its start (correspondingly, when the temperature in the physical annealing process is high), the algorithm will accept most neighbor states, even if these are worse than the current solution. But while the control parameter c_k decreases (correspondingly, when the temperature in the physical annealing process goes down), besides all those neighbors better than the current solution, the algorithm may also accept those worse ones with a certain probability, and this probability of accepting worse solutions decreases with the control parameter c_k going down.

Since the simulated annealing algorithm can give a simple local search technique the possibility (with a minority of changes) to escape from local optima, we exploit this advanced search technique to find a global optima for the combinatorial optimization problems. In fact, the simulated annealing algorithm in its formulation converges with probability 1 to a globally minimal state in either one of the following two cases[19]:

1. An infinite number of state transitions for each value of the control parameter (virtual temperature) c_k and $\lim_{k \rightarrow \infty} c_k = 0$ (the homogeneous algorithm).
2. One state transition for each value of the control parameter c_k and c_k goes to zero not faster than $O[\log k]^{-1}$ (the inhomogeneous algorithm).

However, it is also possible to combine the homogeneous algorithm with the inhomogeneous one by reducing the control parameter c_k slower than $O([\log k]^{-1})$ and keeping a finite number of state transitions at each value of the control parameter c_k .

Although the simulated annealing algorithm is theoretically an optimization algorithm, the asymptotic convergence to the minima can only be approximated in practice. The number of state transitions for each control parameter value c_k , for example, must be finite and $\lim_{k \rightarrow \infty} c_k = 0$ can only be approximated in a fi-

nite number of values for c_k . Therefore, the approximation algorithm is no longer guaranteed to find a global minima with probability 1.

In the practical environment, the simulated annealing process is usually implemented as a sequence of finite homogeneous Markov chains¹ generated at decreasing values of the control parameter c_k . This type of implementation depends on two sets of parameters: the length of the different chains, which can have constant length or a length that depends on the chain, and the control parameter c_k , which is generally defined by an initial value c_0 and a recurrence relation defining c_k as a function of c_{k-1} . These two sets of parameters are not independent of one another. If the control parameter decreases quickly, the homogeneous Markov chains have to greatly increase in length; otherwise, the difference between two stationary distribution² is very small, so few iterations are necessary to stabilize the chain.[2]

More specifically, this sequence of finite homogeneous Markov chains can be specified through a *cooling schedule* in terms of:

- the initial value c_0 of the control parameter c_k ,
- the final value c_f of the control parameter c_k (stop criterion),
- the length of homogeneous Markov chains, and
- a rule for changing the current value c_k of the control parameter into the next one c_{k+1} .

¹A Markov chain is a Markov process x_n having a countable number of states a_i and is specified in terms of its *state probabilities* $p_i[n] = P\{x_n = a_i\}$ $i = 1, 2, \dots$ and the *transition probabilities* $\pi_{ij}[n_1, n_2] = P\{x_{n_2} = a_j | x_{n_1} = a_i\}$. A Markov chain x_n is called *homogeneous* if the transition probability π_{ij} is invariant to a shift of the origin but the state probabilities $p_i[n]$ might depend on n . [22]

²If the limiting values $\lim_{n \rightarrow \infty} p_i[n]$ of the state probabilities exist and are independent of the initial probability vector, we say these homogeneous Markov chains have *stationary* states and their state probabilities are called *stationary* distributions. [14]

4.3 A SA-based Solution to OBP Problem

Generally speaking, the application of the *simulated annealing* algorithm to a combinatorial optimization problem presuppose the definition of:

- a way to represent states,
- a cost,
- a generation mechanism for state transitions, and
- a cooling schedule.

To solve the optimal band partitioning problem in the *self-sizing* network via the *simulated annealing* algorithm, The state space S , the cost function f , the generation mechanism N for state transitions and the cooling schedule are defined as follows[13]:

- *State space*: the state space S for OBP problem is the set of all feasible solutions, namely the network capacity partitioning and traffic routing configuration, $s \in S = \{C_{bj}\}$.
- *Cost function*: the objective function is the cost function $f(s)$ where $s \in S$, i.e. $f(s) = \sum_{b \in D} \sum_{j \in J} C_{bj} w_{bj}$. Note that: C_{bj} and w_{bj} represent the assigned capacity and the cost of basic bandwidth unit for band b in link j , respectively.
- *Generation mechanism*: Randomly choose two SD pairs; for each SD pair, change the route to the alternative one. The move is restricted so that the new solution $\tilde{s}$ is still feasible for the problem. And all of these new solutions $\tilde{s}$ construct the neighborhood $N(s)$ of a solution s .
- *A cooling schedule*:
 1. The initial value c_0 of the control parameter should be determined in such a way that virtually all state transitions are accepted. Thus, one may

first choose a large value for c_0 and perform a number of state transitions. If the *acceptance ratio*, defined as the number of the accepted transitions divided by the number of proposed transitions, is less than a given value (e.g. 1.0), one may double the current value of c_0 and continue this procedure until the observed *acceptance ratio* is larger than or equal to that given acceptance ratio.

2. The final value c_f of the control parameter corresponds to the last homogeneous Markov chain, in which the difference between the maximum and minimum state costs can be achieved by one single cost change incurred in one accepted state transition.
3. The length of the homogeneous Markov chain at each value of the control parameter is determined when the *stationary* of the homogeneous Markov chain is reached. The condition to check for *stationary* has been presented in the literature[15].
4. A decrement rule for the control parameter is given in the literature[15] as follows: $c_{k+1} = c_k \cdot e^{-\lambda \cdot c_k / \sigma_k}$, where $\lambda \leq 1$ and σ_k is the standard deviation of the cost distribution at the control parameter c_k and can be obtained when the homogeneous Markov chain at the control parameter c_k reaches *stationary*.

The above algorithm based on *simulated annealing* to solve OBP problem can be summarized as the following pseudo code:

```

010    initialize  $c_0$  and  $s_0$ ;
020     $k := 0$ ;
030     $s := s_0$ ;
040    repeat
050        repeat
060            begin
070                generate  $\tilde{s}$  from  $N(s)$ ;

```



```

080                if  $e^{-(f(\tilde{s})-f(s))/c_k} > \text{random}[0, 1]$  then  $s := \tilde{s}$ ;
090                else  $s := s$ ;
100            end
110        until stationary reached
120        calculate  $\sigma_k$ ;
130         $k := k + 1$ ;
140        calculate  $c_{k+1}$ ;
150    until the stop criterion valid

```

It deserves to be mentioned that the above solution to OBP problem is based on the premise that the whole network is flat and centralized. More specifically, All enhanced switching nodes periodically send traffic and network state information to the sole central server, whereas the central server determines the whole network partitioning and traffic routing and sends all configuration information to all enhance switching nodes in time.

Here, we present an OBP problem instance and its solutions referred in the literature[13], where the *simulated annealing* technique is firstly applied as a fast analysis and synthesis algorithm to solve the OBP problem of the *self-sizing* network.

The *self-sizing* network, examined in the literature[13], is shown on Figure 4.1. It involves one central server, four enhanced switching nodes and seven duplex links. Each duplex link consists of two reverse simplex links whose individual physical capacity is 155 (*Mbps*) (OC-3). Assume that two reverse simplex links within the same duplex link have the same bandwidth unit cost for all different traffic classes. Table 4.1 gives the bandwidth unit cost w_{bj} of every simplex link j shown in Figure 4.1.

Thus the set of simplex links $J = \{01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14\}$ and the set of the bandwidth unit cost $W = \{2.6, 2.5, 2.4, 2.5, 2.5, 2.3, 2.4, 2.6, 2.5, 2.4, 2.5, 2.5, 2.3, 2.4\}$. Note that the bandwidth unit cost w_{bj} , which could be associated with the length of links, consists of the bandwidth usage cost

Figure 4.1: A practical self-sizing network

Table 4.1: The bandwidth unit cost of simplex links in single-size network

simplex link	link no.(j)	simplex link	link no.(j)	bandwidth unit cost (w_{bj})
server_1→ node_2	01	node_2→ server_1	08	2.6
server_1→ node_3	02	node_3→ server_1	09	2.5
node_2→ node_3	03	node_3→ node_2	10	2.4
node_2→ node_4	04	node_4→ node_2	11	2.5
node_2→ node_5	05	node_5→ node_2	12	2.5
node_3→ node_4	06	node_4→ node_3	13	2.3
node_3→ node_5	07	node_5→ node_3	14	2.4

and the port processing cost.

After setting up the whole physical network, the authors of the literature[13] further assume that three types of traffic, i.e. voice, video and data, traverse simultaneously this network. To meet different requirements for QoS, this network is logically divided into three virtual subnetworks. That is to say each simplex link has to be partitioned into three virtual bands: band '1' for voice traffic, band '2' for video traffic and band '3' for data traffic. Thus the set of traffic classes

$B = \{1, 2, 3\}$ and the effective bandwidths of voice, video and data traffic are given 0.99 (*BBU*), 16.77 (*BBU*) and 3.25 (*BBU*), respectively[35].

Table 4.2: The workload of each traffic class b between each active SD node pair p

the active SD node pair(p)	voice (1)	video (2)	data (3)
server_1 $\rightarrow$ node_2	101	60	32
server_1 $\rightarrow$ node_3	118	34	13
server_1 $\rightarrow$ node_4	135	30	40
server_1 $\rightarrow$ node_5	104	40	29
node_2 $\rightarrow$ node_3	118	76	14
node_2 $\rightarrow$ node_4	114	130	102
node_2 $\rightarrow$ node_5	107	70	41
node_3 $\rightarrow$ node_4	108	25	16
node_3 $\rightarrow$ node_5	99	16	17
node_4 $\rightarrow$ node_5	123	36	29

There are ten active SD node pairs in the network shown on Figure 4.1. Between each pair of active SD nodes there exist all three different types of traffic. Table 4.2 gives the number v_{pb} of all connections in traffic classes b between each active SD node pair p . For the traffic between each SD node pair, either the shortest path or the second shortest path might be chosen to deliver them.

Then, the approximation algorithm mentioned in section 4.3 on page 39 is implemented using C++ at the platform of UNIX. Subsequently, a cooling schedule is determined via $c_0 = 5000$ and $\lambda = 0.7$. Eventually, the optimal link partitioning and traffic routing can be figured out within 0.1 (*s*) when the approximation algorithm is applied to the case described above on the platform of Sun Ultra 5. Thus, the approximation algorithm is not only effective but also feasible in the above *self-sizing* network.

Herein, the feasibility of the approximation algorithm refers to whether or not its execution time is able to meet the requirement for the timeliness of the *self-sizing* mechanism. However, the execution time of the approximation algorithm depends on the OBP problem size that is concerned about the number of both links and active SD node pairs³ of the *self-sizing* network.

When the network scales up, both links and the active source-destination node pairs might increase exponentially. *Is the fast analysis and synthesis algorithm based on the simulated annealing technique feasible in the large-scale self-sizing network?* In order to answer this question, we must examine how well the approximation algorithm solve the large-scale OBP problem in the *self-sizing* network.

4.4 A Large-scale OBP Problem

Since any implementation of the *simulated annealing* algorithm is a finite approximation to the asymptotic annealing process, the performance analysis of the approximation algorithm should concentrate on the following two quantities:

1. the quality of the final solution obtained by the approximation algorithm, i.e. the difference in cost value between the final solution and a globally minimal configuration;
2. the execution time required by the approximation algorithm to obtain the final solution.

Traditionally, i.e. for deterministic algorithms, one distinguishes between two different types of performance analysis: worst-case analysis and average-case analysis. For a given probability distribution of problem instances, the worst-case analysis is concerned with upper bounds on the quality and the execution time

³The source-destination node pair is called *active* if there exists traffic between this source-destination pair

whereas the average-case analysis is concerned with expected values of the quality and the execution time.

An additional probabilistic aspect is added to the aforementioned classification because the *simulated annealing* algorithm is probabilistic. Besides the probability distribution over the set of problem instances, there is also a probability distribution over the set of possible solutions, which the algorithm can obtain for a given problem. Thus, in an average-case analysis, the average can refer to either the set of problem instances or the set of solutions of a given problem instance.

In this thesis, we only consider the average-case analysis over the set of solutions for a given problem instance since this analysis method enables us to have direct understanding on how the scale of OBP problem influences the performance of the approximation algorithm. To maintain the analysis consistency, we start up from the scenario discussed on section 4.3.

To clarify how the execution time behaves when the network scales up, on the one hand, we fix the number of links in the network and increase the number of the active SD node pairs; on the other hand, we keep the number of the active SD pairs and their distribution and increase the number of links[28].

4.4.1 Active SD node pairs

In the *self-sizing* network shown on Figure 4.1, we gradually increase the number of active SD node pairs. We assume that there exist only the first **two** active SD node pairs from those ten pairs shown on Table 4.2, randomly run the approximation algorithm over these traffic loads more than 30 times on a Sun-Ultra 5 workstation and obtain the average execution time (*ms*) of this situation.

Then, we assume that only the first **three** active SD node pairs from ten pairs shown on Table 4.2, repeat the above processing and get the average execution time of this new situation. And so on and so forth, we keep increasing the number of active SD node pairs and acquire a series of the average execution time values,

as shown in Table 4.3, corresponding to different number of active SD node pairs existing in the network illustrated on Figure 4.1.

Table 4.3: The average execution time vs the number of active SD node pairs

SD node pairs	2	3	4	5	6	7	8	9	10
execution time	3.3	8.7	18.8	33.2	52.4	86.4	116.2	132.9	205.1

Figure 4.2: The average execution time vs the number of the active SD node pairs

From these data in the Table 4.3 and the curve shown in Figure 4.2, we can see that the average execution time increases faster than $O[n]$ when the number n of active SD node pairs grows. This implies that the performance of this approximation algorithm in terms of the execution time deteriorates dramatically when the network scales up.

4.4.2 Quantity of links

In order to inspect how the number of links in the *self-sizing* network affects the average execution time of the approximation algorithm, we symmetrically double and quadruple the size of the network shown on Figure 4.1 and assume that two symmetric links have the same bandwidth unit cost.

Figure 4.3: A double-size self-sizing network

Figure 4.4: A quadruple-size self-sizing network

The double-size network that consists of two symmetric single-size networks connected via an OC-12 duplex link, is shown on Figure 4.3. Likewise, the quadruple-size network that includes two symmetric double-size networks connected through two OC-12 duplex links is illustrated on Figure 4.4. From these topologies of single/double/quadruple-size networks, we can see that the whole single-size network is exactly one part of the double-size network and the whole double-size network is also one part of the quadruple-size network. Thus, we only need to know the bandwidth unit cost w_{bj} that is given in Table 4.1 and Table 4.4.

Table 4.4: The bandwidth unit cost of simplex links in quadruple-size network

simplex link	link no. (j)	simplex link	link no. (j)	bandwidth unit cost (w_{bj})
server_1 $\rightarrow$ node_6	15	node_6 $\rightarrow$ server_1	16	3.2
node_6 $\rightarrow$ node_7	17	node_7 $\rightarrow$ node_6	24	2.6
node_6 $\rightarrow$ node_8	18	node_8 $\rightarrow$ node_6	25	2.5
node_7 $\rightarrow$ node_8	19	node_8 $\rightarrow$ node_7	26	2.4
node_7 $\rightarrow$ node_9	20	node_9 $\rightarrow$ node_7	27	2.5
node_7 $\rightarrow$ node_10	21	node_10 $\rightarrow$ node_7	28	2.5
node_8 $\rightarrow$ node_9	22	node_9 $\rightarrow$ node_8	29	2.3
node_8 $\rightarrow$ node_10	23	node_10 $\rightarrow$ node_8	30	2.4
server_1 $\rightarrow$ node_11	31	node_11 $\rightarrow$ server_1	32	3.2
node_6 $\rightarrow$ node_16	33	node_16 $\rightarrow$ node_6	34	3.2
node_11 $\rightarrow$ node_12	35	node_12 $\rightarrow$ node_11	42	2.6
node_11 $\rightarrow$ node_13	36	node_13 $\rightarrow$ node_11	43	2.5
node_12 $\rightarrow$ node_13	37	node_13 $\rightarrow$ node_12	44	2.4
node_12 $\rightarrow$ node_14	38	node_14 $\rightarrow$ node_12	45	2.5
node_12 $\rightarrow$ node_15	39	node_15 $\rightarrow$ node_12	46	2.5
node_13 $\rightarrow$ node_14	40	node_14 $\rightarrow$ node_13	47	2.3
node_13 $\rightarrow$ node_15	41	node_15 $\rightarrow$ node_13	48	2.4
node_11 $\rightarrow$ node_16	49	node_16 $\rightarrow$ node_11	50	3.2
node_16 $\rightarrow$ node_17	51	node_17 $\rightarrow$ node_16	58	2.6
node_16 $\rightarrow$ node_18	52	node_18 $\rightarrow$ node_16	59	2.5
node_17 $\rightarrow$ node_18	53	node_18 $\rightarrow$ node_17	60	2.4
node_17 $\rightarrow$ node_19	54	node_19 $\rightarrow$ node_17	61	2.5
node_17 $\rightarrow$ node_20	55	node_20 $\rightarrow$ node_17	62	2.5
node_18 $\rightarrow$ node_19	56	node_19 $\rightarrow$ node_18	63	2.3
node_18 $\rightarrow$ node_20	57	node_20 $\rightarrow$ node_18	64	2.4

Table 4.5: The average execution time of different network sizes

SD node pairs	2	3	4	5	6	7	8	9	10
single size	3.3	8.7	18.8	33.2	52.4	86.4	116.2	132.9	205.1
double size	6.8	17.2	42.5	69.7	108.6	171.8	238.8	269.2	414.2
quadruple size	13.2	35.7	82.6	149.2	205.7	351.9	482.6	548.5	839.4

As with before, we gradually increase the number of active SD node pairs from the first two pairs to the first ten pairs shown on Table 4.2, solve this OBP problem over every specific number of the active SD node pairs more than 30 times using the same seeds as used in the single-size network scenario and obtain the average execution time of these three scenarios, as shown on Table 4.5.

Figure 4.5: The average execution time of different network sizes

From the data in the Table 4.5 and the curves shown in Figure 4.5, we can see that the average execution time increases linearly with the number of the links since

$$T_{running} \approx K * L * l * \tau. \quad (4.6)$$

K : the number of homogeneous Markov chains, determined by the decrement

process of the control parameter c_k (the cooling rule).

L : the number of state transitions (i.e. the chain length) for the homogeneous Markov chain to reach *equilibrium* at some value of the control parameter c_k .

l : the total number of links in the network.

τ : the time needed to check the physical capacity constraint of one link.

For these three single/double/quadruple-size scenarios, their individual decrement processes of the control parameter c_k are all the same one. Furthermore, their individual homogeneous Markov chains based on the identical value of the control parameter c_k have the same length. Nevertheless, the required time for state transitions in their individual homogeneous Markov chains is different and depends on the total number of links in the network since this approximation algorithm must check the capacity constraints of each link before state transitions. Thus, the larger the network is, the more the links are and the longer the time needed for state transitions is.

Also, we can verify that the average execution time grows faster than $O[n]$ when the number n of active SD node pairs increases regardless of the size of the network from curves shown in Figure 4.5

4.5 A Hierarchical Self-Sizing Framework

Since the scale-up of the network size dramatically worsens the performance of the approximation algorithm in terms of the execution time, it is possible for the fast analysis and synthesis algorithm in *centralized* mode unable to solve the large-scale OBP problem within the required time. We could split the whole network into some subnets so that each subset has less links and active SD node pairs⁴ since the number of both links and active SD node pairs in the network have a negative impact on the execution time of the approximation algorithm.

⁴Not all active SD node pairs are global, thus the number of the active SD node pairs in each subnet is always less than or equal to the number of the active SD node pairs in the whole network

Then, we simultaneously apply the approximation algorithm locally to each subnet so that the execution time can be reduced significantly due to the parallel processing and the down-sized scale of OBP problem. This application mode of the approximation algorithm is called *distributed* corresponding to the *centralized* mode in which the approximation algorithm is applied to the whole network. Of course, we may not reach the global optimum in this case. Thus, we have to investigate how the *distributed* mode of the analysis and synthesis algorithm saves the execution time and deteriorates the cost value compared with the *centralized* mode.

Next, we still consider the network topology shown on Figure 4.4 and assume some globally distributed traffic loads given in Table 4.6. Then, we apply the approximation algorithm in *centralized* mode to the whole network and get the global optimum (79165.7) as well as the average execution time (1690.9 ms) to reach the global optimum.

Table 4.6: The workload of each traffic class between each active SD node pair

SD node pair	voice (1)	video (2)	data (3)
node_10→ node_4	51	30	16
node_9→ node_5	59	17	7
node_10→ node_15	78	15	20
node_9→ node_15	52	20	15
node_19→ node_14	59	38	7
node_20→ node_15	57	65	51
node_5→ node_14	54	35	22
node_2→ node_14	54	13	8
node_7→ node_20	50	8	9
node_8→ node_20	62	18	15

To apply this *distributed* approach to solve the OBP problem in the quadruple-size network shown on Figure 4.4, we propose a hierarchical *self-sizing* framework

Figure 4.6: A hierarchical self-sizing framework

shown on Figure 4.6 . We divide the global network into two domains: the first domain comprises one sub-network containing 4 servers (1, 6, 11, 16), 8 simplex OC-12 links and ten pairs of active SD nodes; the second domain consists of 4 similar sub-networks, which are sub-network_1 including 5 nodes (1, 2, 3, 4, 5), 14 simplex OC-3 links and four pairs of active SD nodes; sub-network_2 including 5 nodes (6, 7, 8, 9, 10), 14 simplex OC-3 links and six pairs of active SD nodes; sub-network_3 including 5 nodes (11, 12, 13, 14, 15), 14 simplex OC-3 links and six pairs of active SD nodes and sub-network_4 (16, 17, 18, 19, 20), 14 simplex OC-3 links and four pairs of active SD nodes.

By this way, we can acquire two additional benefits as follows:

1. reducing the transmission overhead to save network resources since we only need to send the information of both traffic load and network state to the corresponding servers of the subnetworks;
2. decreasing the number of active SD node pairs in the subnetworks to save the execution time as long as we break up the whole network into a hierarchical

framework intelligently because not all of traffic is global.

We concurrently apply the approximation algorithm onto two domains/five individual sub-networks, respectively. And we sum the optima (cost values) of all sub-networks to be 82620.35, greater than the global optimum (79165.7), but the average execution time is 573.8(*ms*), far less than the average execution time (1690.9*ms*) in *centralized* mode. It means that the optimum in *distributed* mode deteriorates only 4.4% but the execution time decreases 4 times.

Obviously, this result is quite reasonable because the amount of links decreases dramatically from 64 links to 8 links for the sub-network in the first domain even though the number of active SD node pair do not decrease; for those sub-networks in the second domain, not only does the number of active SD node pairs reduce from 10 to 4/6, but also the amount of links lessens significantly from 64 to 14. Eventually, the average execution time that depends on the number of links as well as the amount of active SD node pairs decreases spectacularly.

In summary, the average execution time of the approximation algorithm rises approximately faster than $O[n]$ when the number n of active SD node pairs increases. On the other hand, the average execution time also grows proportionally to the number of links. However, we can lessen the average execution time significantly but only slightly deteriorate the optima using the *distributed* mode instead of the *centralized* mode through a *hierarchical* self-sizing framework.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In this thesis, we have addressed two critical issues of the *self-sizing* mechanism. One deals with the estimation of real-time traffic load through on-line measurement of effective bandwidth so that network resources allocated to these real-time traffic can be adaptively adjusted (i.e. self-sizing); The other one is how to process the real-time traffic and network state information via the analysis and synthesis algorithm in a large-scale network so that traffic flows can be balanced through whole network within the required time framework.

In order to implement the *self-sizing* mechanism, it is essential to measure effective bandwidth instantly and precisely. There have been many studies on the measurement of EB. However, the approach proposed by Courcoubetis *et. al* in [32] might be the most promising because it is applicable for all traffic and doesn't need any *a priori* knowledge of traffic. Although it is based purely on measurements, this approach is still too slow to meet the real-time requirement of on-line measurements due to its computation complexity.

Therefore, we start off from the expression (3.10) presented by R.J.Gibbens in [12] to measure effective bandwidth as well as estimate the real-time traffic load (i.e. the consumed link capacities) for resource allocation or CAC mechanisms. Then,

we introduce the concept of an “epoch” to reduce the computation complexity of expression (3.11) by at least $2m$ times. Subsequently, we choose a sample trace whose length in time is exactly J (epoch) and a time scale K (epoch), thus, we eliminate all of multiplications to reduce the computation of expression (3.11) once more. After that, we propose a recursive algorithm to cut down the complexity of expression (3.11) by $J-K$ times so that we only need one time of exponent calculation, addition and subtraction, respectively. However, we prefer the small window width J of the sample trace because of calculation delay and potential numerical overflow.

For the measured EB value to be more precise, we add another recursive algorithm to take the whole available traffic trace into account. So, we eventually formulate a novel Double Recursive algorithm with Double Moving Window (DRDMW) to accelerate the EB estimation and prevent the potential numerical overflow. Furthermore, simulation results show that our proposed DRDMW algorithm is effective.

After the real-time traffic load is reported on-line, the analysis and synthesis algorithm is used to balance traffic flows. Hao *et. al* modeled this process into an optimal link partitioning (OBP) problem and solve it using the simulated annealing algorithm in [13]. We investigate the performance of the simulated annealing algorithm in the large-scale network and find its execution time increases faster than $O[n]$ when the number of n of active SD node pairs grows; in addition its execution time also increases linearly with the number of the network links.

In order to prevent the violation of timeliness of the *self-sizing* mechanism, we propose a hierarchical framework to split a large-scale OBP problem in the case of large-scale network into some small-scale OPB problems of subnets and simultaneously apply the simulated annealing algorithm to solve the down-sized OBP problems. Simulation results show our proposed hierarchical *self-sizing* framework can dramatically reduce the execution time of the simulated annealing algorithm but only slightly deteriorate the performance in terms of the cost.

5.2 Future Research

Regarding the on-line measurement of effective bandwidth, we have justified the proposed DRDMW algorithm through the synthesized traffic trace resembling fractional Gaussian noise in this thesis. Furthermore, we need to investigate the following:

- Given the realistic traffic trace measured in practice, how well is the accuracy of the DRDMW, compared with the conventional method proposed by Gibbens when we search the particular values of 's' and 't', estimate the EB of traffic as well as measure the consumed link capacities?
- If we would regard the aggregate existing traffic as a single traffic, then the minimum consumed link capacities such that the maximum CLR can be guaranteed will be the EB of this traffic. Therefore, is it possible for us to search the particular values of 's' and 't' and estimate the EB of the aggregate traffic only in terms of the aggregate traffic trace? If so, we will not only have no need to preserve the per-flow state required in equation (3.14) and reduce the computation complexity significantly but also improve the precision of the EB measurement.

Bibliography

- [1] <http://www-st.inf.tu-dresden.de/aquila/>
- [2] Azencott, R. SIMULATED ANNEALING: Parallelization Techniques. pp. 47-49, JOHN WILEY & SONS, INC. 1992, ISBN 0-471-53231-2.
- [3] Botvich, D.D. and Duffield, N.G. Large deviations, the shape of the loss curve, and economies of scale in large multiplexers. *Queuing Systems*, 20:293-320, 1995.
- [4] Courcoubetis, C., Kelly, F.P. and Weber, R. Measurement-based usage charges in communications networks. Technical Report 1997-19, Statistical laboratory, University of Cambridge, 1997. To appear in *Operations Research*.
- [5] Courcoubetis, C. and Weber, R. Buffer overflow asymptotic for a switch handling many traffic sources. *J. Appl. Prob.*, 33:886-903, 1996.
- [6] Duffield, N.G., Lewis, J.T., and O'Connell, N. Predicting Quality of Service for Traffic with Long-Range Fluctuation. *IEEE ICC'95*, pp. 473-477, 1995.
- [7] Duffield, N.G., Lewis, J.T., O'Connell, N., Russell, R., and Tomez, F. Entropy of ATM Traffic Streams: A tool for Estimating QoS Parameters. *IEEE Journal on Selected Areas in communications*, vol. 13, no. 6, pp. 981-990, August 1995.
- [8] deVeciana, G., Kesidis, G. and Walrand, J. Resource Management in Wide-Area ATM Networks Using Effective Bandwidths. *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1081-1090, August 1995.

- [9] Duffield, N.G. and O'Connell, N. Large deviations and overflow probabilities for the general single-server queue with applications In Math, Proc Cambridge Philos. Soc. 1995, pp363-375.
- [10] Elwalid, A.I. and Mitra, D. Effective bandwidth of general Markovian traffic sources and admission control of high speed networks. IEEE/ACM Trans. on Networking, 1(3):329-343, October 1993.
- [11] Falkner, M., Tartarelli, S., Devetsikiotis, M. and Lambadaris, I. Measuring Effective Bandwidths and the concept of Empirical Effective Bandwidths. Technical Report, Carleton University, August 1999, SCE REPORT 99-08.
- [12] Gibbens, R.J. Traffic Characterization and Effective Bandwidths for Broad-band Networks Traces, in Stochastic Networks: Theory and Application, Royal Statistical Society Lecture Note Series, F.P. Kelly, S. Zachary, and I.B. Ziedins, Eds. vol. 4, pp. 169-179. Oxford University Press, 1996 ISBN 0-19-852399-8
- [13] Hao, Q., Tartarelli, S. and Devetsikiotis, M. Self-sizing and Optimization of High-Speed Multi-service Networks. Proceedings of IEEE GLOBECOM 2000, Vol 3, pp. 1818-1823.
- [14] Ng, C.H. Queueing Modelling Fundamentals. pp. 65-80 JOHN WILEY & SONS, 1997 ISBN 0-471-96819-6.
- [15] Huang, M.D., Romeo, F. and Sangiovanni-Vincentelli, A. An Efficient General Cooling Schedule for Simulated Annealing. Proceedings of IEEE ICCAD'86, Santa Clara, CA, Nov. 1986, pp.381-384.
- [16] Kelly, F.P. Notes on Effective Bandwidths. Stochastic networks: Theory and Applications, Royal Statistical Society lecture Notes Series, volume 4, pages 141-168. Oxford University Press, 1996.

- [17] Keshav, S. An Engineering Approach to Computer Networking: ATM networks, the Internet, and the Telephone Network. ADDISON-WESLEY, Jan.15,1997, pp. 466.
- [18] Kesidis, G., Walrand, J. and Chang, C.-S. Effective bandwidths for multi-class Markov fluids and other ATM sources. IEEE/ACM Trans. on Networking, 1(3):424-428, August 1993.
- [19] Laarhoven, P.J.M.van and Aarts, E.H.L. Simulated Annealing: Theory and Applications. D.Reidel Publishing Company, 1987.
- [20] Lewis, J.T. and Russell, R. An Introduction to Large Deviation for Teletraffic Engineers. In Performance '96, October 1996. Available from *URL http : /www.stp.dias.ie/APG/dias_apg_pub.html*.
- [21] Papadimitriou, C.H. and Steiglitz, K. Combinatorial Optimization: Algorithm and Complex. Englewood Cliffs, NJ:Prentice-Hall, 1982
- [22] Papoulis A. Probability, Random Variables, and Stochastic Processes. pp. 635-639, Third Edition, McGraw-Hill Inc.
- [23] Park, K., and Willinger, W. Self-Similar Network Traffic and Performance Evaluation. New York, J. Wiley 2000. xiv.
- [24] Paxson, V. Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic. Computer Communications Review 27(5), pp. 5-18, Oct. 1997.
- [25] Saito, H., Nakagawa, S., Kato, Y. and Ohta, S. Innovation of circuit/path operations in ATM networks. NTT Review, vol.8, no. 1, pp. 56-65, 1996
- [26] Semeria, C. Multiprotocol Label Switching-Enhancing Routing in the New Public Network. White Paper, www.juniper.net, Juniper Networks.

- [27] Semeria, C. Supporting Differentiated Service Classes in Large IP Networks. White Paper, www.juniper.net, Juniper Networks.
- [28] Shen, W. and Devetsikiotis, M. A Self-sizing Framework for Resource Allocation in Label-Switched Networks. to be submitted to IEEE International Conference on Computer Communications and Networks, 11th. Miami, Florida, USA, Oct., 2002.
- [29] Shen, W. and Devetsikiotis, M. On-line Measurement of Effective Bandwidth for Resource Allocation in Label-Switched Networks. to be submitted to IEEE International Conference on Communications, Alaska, USA, May 2003.
- [30] Shioda, S. *et al.* Self-Sizing Network: A New Network Concept Based on Autonomous VP Bandwidth Adjustment. Proc. Int'l. Teletraffic Cong., ITC15, Washington, DC, June 1997, pp. 997-1006.
- [31] Simonian, A. and Guibert, J. Large deviations approximations for fluid queues fed by a large number of on/off sources. IEEE J.Select. Areas Commun., 13(7):1017-1027, August 1995.
- [32] Courcoubetis, C., and Siris, V.A. Procedures and tools for analysis of network traffic measurements. Accepted for publication in Performance Evaluation Journal, Submitted September 2000.
- [33] Tartarelli, S., Falkner, M., Devetsikiotis, M., Lambadaris, I. and Giordano, S. Empirical Effective Bandwidths. Proceedings of IEEE GLOBECOM 2000, Vol 1, pp. 672-678.
- [34] <http://www.ist-tequila.org/>
- [35] Yang, J. and Devetsikiotis, M. On-line Estimation, Network Design and Performance Analysis with Effective Bandwidths. in Proceedings of International Teletraffic Conference, 17th. Brazil, Dec, 2001.

- [36] Yan, J. Adaptive Configuration of Elastic High-Speed Multi-class Networks. IEEE Communications Magazine, May 1998, pp. 116-120.
- [37] Yan, J. and Beshai, M. Designing an ATM-Based Broadband Network: An Overview. GLOBECOM '95, Singapore, Nov. 1995.