DYNAMIC BANDWIDTH MANAGEMENT TO MAXIMIZE USER SATISFACTION DEGREE ON MULTIPLE MPLS PATHS

WU ZHENG

NATIONAL UNIVERSITY OF SINGAPORE

2005

DYNAMIC BANDWIDTH MANAGEMENT TO MAXIMIZE USER SATISFACTION DEGREE ON MULTIPLE MPLS PATHS

WU ZHENG

(*B.Sc., Nanjing University, China*)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgment

I would like to express my gratitude to my supervisor Dr. Yuming Jiang for his seasoned guidance. His ideas and suggestions helped a lot in my research work. He also gave me a lot of kind help when I came to Singapore first time.

I appreciate my supervisor Dr. Minh Hoang Nguyen, for his continuous guidance and encouragement in my postgraduate studies. Thanks to his amazing experience, he helped me a lot when I struggled in finding a research point. His advice helped me to move towards the right direction.

I am extremely grateful to my supervisor Dr. Qinghe Yin who very kindly advised and helped me for this thesis.

It is my pleasure to work with my colleague: Yixin Huang, Qijie Huang, Haiming Xiao, and many other friends. Their ingenious mind gave me a lot of good suggestions and ideas when I discussed with them. They also brought me a enjoyable research life.

I dedicate this thesis to my parents for their continuous support in my life.

Contents

Summary

Resource allocation and management is an important part of traffic engineering for today's networks. This thesis presents a dynamic bandwidth management scheme for traffic on multiple MPLS paths. We emphasize that the degree of end user's satisfaction is the ultimate goal of traffic engineering. In our scheme we propose a utility function system to describe the relationship between end user's satisfaction degree and resources allocated to that user. With the system we can define a utility function for an application based on its QoS requirements.

When utility functions are given, maximizing network's total utility becomes a constrained optimization problem. We propose a heuristic approach to solve this problem. With the QoS information brought by the utility function, admission control is first executed. After making a selection from multiple paths, a load balancing algorithm based on Gradient-Projection method is used to maximize total utility. Moreover, priority and preemption policy is also considered in our heuristic approach. With the constraint-based routing (CR) function provided by MPLS, we can easily implement our scheme in MPLS network.

Simulation results show that our scheme can dynamically manage the allocation of resources when traffic load changes, and keep QoS requirements

of applications satisfied. When compared with current multiple path routing algorithms our proposed heuristic algorithm performs better in both blocking probability and total utility.

Keywords: Utility Function, Traffic Engineering, Bandwidth Management, Load Balancing, MPLS

List of Figures

Abbreviations

FCFS:	**First Come First Served**
FEC:	**Forwarding Equivalence Class**
LDP:	**Label Distribution Protocol**
LIB:	**Label Information Base**
LSP:	**Label Switched Path**
LSR:	**Label Switching Router**
MPLS:	**MultiProtocol Label Switching**
QoS:	**Quality of Service**
RD:	**Random multiple paths routing algorithm**
SW:	**Shortest-Widest first multiple paths routing algorithm**
WS:	**Widest-Shortest first multiple paths routing algorithm**

Chapter 1

Introduction

1.1 A Brief Introduction to Internet Traffic Engineering

In recent years, there has been an explosive growth in the development and deployment of Internet applications. With more and more applications transmitting and receiving audio or video contents over the Internet, the Internet traffic load grows at an exponential speed, which makes many network devices such as routers are becoming bottlenecks in the Internet. Also, with the emergence of more and more new Internet applications with different transmitting requirements such as upper delay bound and packet loss ratio bound, it is becoming the Internet support current and future applications effectively is becoming a great challenge.

Traffic Engineering is considered as one component of an autonomous

system required to support applications effectively [3]. The task of traffic engineering is performance evaluation and performance optimization of operational IP network. The performance of the network seen by end users is the crucial point and should be considered throughout the traffic engineering process. In this thesis, the performance we talk about is the satisfaction degree of end users, or named as *utilities* [22], which will be described in detail in Chapter 2. Performance optimization normally includes two parts: capacity management and traffic management. Traffic management deals with the regulating of traffic flows. Capacity management includes capacity planning, routing control, and resource management. Since the operating environments of contemporary Internet networks are dynamic, performance optimization has two sub-processes: real-time network optimization and static network planning. In this thesis we focus on the capacity management part in real-time network optimization sub-process: allocation and dynamic management of link bandwidth, which is one kind of network resources.

1.2 A Brief Introduction to Constraint-Based Routing and MPLS

One of the main traffic engineering tools is constraint-based routing. Constraints may include capacity constraints and Quality of Service (QoS) constraints etc. Constraint-based routing is defined as a class of routing systems for computing feasible network paths subject to a traffic description and a

set of constraints. [4][18][3]

MultiProtocol Label Switching (MPLS) [34] is an advanced forwarding technology. It extends the Internet routing model and enhances packet forwarding and path control. One of its advantages is that it supports the setup of constraint-based routed label switched path (CR-LSP), with which we can easily implement constraint-based routing.

The basic idea of MPLS is label switching on IP layer. Routers in an MPLS domain are named label switching routers (LSRs). IP packets are classified into forwarding equivalence classes (FECs) based on some factors (e.g., destination address, QoS requirements etc) at domain ingress. Then packets that in same FEC will be attached with same unique label. These packets with same label will follow one label switched path (LSP) which originates from an ingress and ends at an egress in MPLS domain. LSP is setup with certain signaling protocol. One signaling protocol designed especially for MPLS is Label Distribution Protocol (LDP). Each LSR in MPLS domain maintains one table named Label Information Base (LIB), which is used to map one label to a specified LSP. When a LSR receives a labeled packet, it checks its label and incoming port, then finds its new label and outgoing port from LIB, uses the new label to displace the old one (label switching) then forwards it from the indicated out port (label forwarding).

With LDP we can set up an explicit path by manually configuring LIBs in LSRs along that path. However, explicit routing, whose constraint is the explicit route, is one subset of the more general constraint-based routing. To support CR-LSP with other constraints such as link bandwidth, delay,

hop count etc, researchers extend LDP to CR-LDP [18]. The main idea of supporting CR-LSP is using TLVs (Type/Length/Value) to describe traffic characters and requirements explicitly. The admission control and resource manager components will use those TLVs to manage resources. More details about CR-LDP is defined in [18], and the implementation of CR-LDP used in our work is described in [1].

1.3 Motivation and Research Goal

With more and more different networking applications such as Voice over IP (VoIP) and Video-On-Demand (VOD) emerging today, the networks which treat all applications as conventional data application become out of date. The network that can satisfy the QoS requirements of these applications is in demand. Since the requirements of today's networking applications are various, for each type of application, designing one special networking architecture just for this type of application is a great cost. Moreover, when new application emerges in the future such non-scalable networks may become useless. So a network that can use one common infrastructure to support today's and future various applications is economical and effective. Therefore, using one common infrastructure to support multiple types of traffic flows is our basic motivation of this thesis.

A number of multiple paths routing algorithms and network resource allocation policies have been proposed in literature. However most of them didn't consider the different QoS requirements of different type of applica-

tions. After Shenker proposed the shapes of utility function curves for elastic applications and real-time applications with different QoS requirements in [36], researchers believe that the utility function can map the degree of end-user's satisfaction of an application to QoS requirements, and can be used with some heuristic to maximize total degree of all end-users' satisfaction. However, most of publications dealing with traffic engineering on multiple paths by using utility functions focus on networks with only one type of traffic flows, but the scenario that elastic traffic flows and different kinds of real-time flows exist on a network simultaneously has not been considered. The objective of this thesis is to design a utility function and propose a heuristic approach to maximize total degree of satisfaction for network with different types of traffic flows in it.

1.4 Research Methodology

We first introduce a utility function which accords with Shenker's curves in [36]. With adjusting parameters in the utility function it can support traffic flows with different QoS requirements. Based on the utility function, the admission control heuristic and bandwidth management algorithm for single path is proposed. Our utility function is divided into two parts in this heuristic. The first part is for admission control, and the other part is to maximize total satisfaction degree by using gradient-projection method. When there are multiple paths between one source-destination pair, we combine the heuristic for single path with existing path selection algorithms. We also

design a path selection method to choose the path that can mostly increase the total utility. By involving this path selection heuristic, we can maximize the total utility in multiple paths cases.

A free discrete event simulator ns-2 [37] is used for simulation. With a extension package named MNS [1] ns-2 can simulate constraint-based routing for QoS traffic in MPLS well.

Two assumptions are made in this thesis. Firstly, topology and routing information are stored accurately in each edge router, i.e., edge routers know the multiple paths and their available capability between any given source-destination pair. Secondly, QoS information requested explicitly by applications is exact and truthful.

1.5 Structure of the Thesis

In Chapter 2, a brief review to current utility functions for elastic applications and real-time applications is presented. Then the gradient-projection method is described. At last we introduce current multiple path routing algorithms.

Chapter 3 introduces our proposed utility functions in this thesis. Then the heuristic for bandwidth allocating and managing on single path using our utility function is described. Finally, the result of throughput of each traffic flow and total utility of whole network are demonstrated through simulation.

Chapter 4 extends the heuristic in single path to multiple paths network. Four heuristics with different path selection algorithms for multiple

paths network are described. The blocking probabilities and total utility of these algorithms are evaluated and compared. Both no preemption case and preemption case are considered.

Chapter 5 concludes this thesis and discusses some issues for possible future research.

Chapter 2

Literature Review

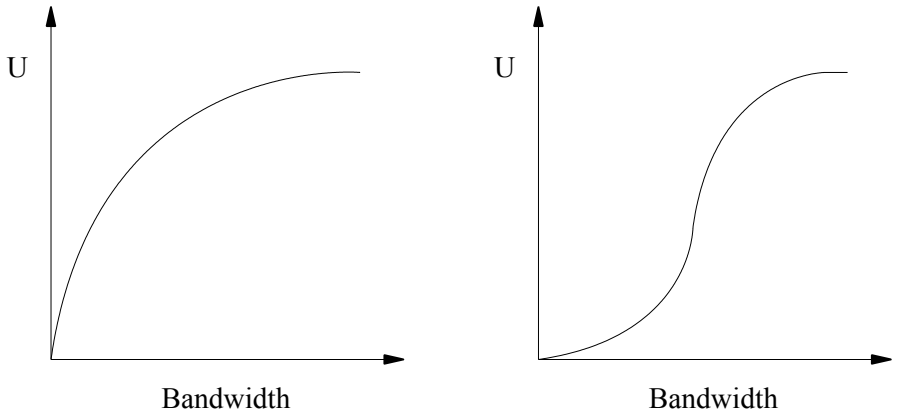
2.1 Existing Utility Functions

One fundamental problem in the Internet design is the allocation of network resources. Recently many research works have been done for traffic engineering and management to use network resources efficiently in order to provide better services. But how could a network provide "better" services to users? Before we talk about any technical term we need to understand what kind of services is better. When "better" is referred, there must be some criteria to evaluate. The Internet was designed to meet the needs of users, so that the evaluative criteria are not link utilization, packet-loss ratio or throughput, but "how happy does this architecture make the users" [36]. For example, load balancing is often mentioned in recent research works, but why is it needed? We may answer that's because load balancing can reduce congestion. But why congestion control is needed? That's because congestion

control can lead to relative small delay and low packet loss ratio. But why small delay and low packet loss ratio are expected? At last we noticed that's because users ask for their applications have small delay and low packet loss. So the final objective of traffic engineering is to satisfy our end-users as best as we can, or say, to maximize the total degree of user satisfaction.

The degree of user satisfaction can be translated into some quality of service (QoS) such as bandwidth or delay requirements by a utility function $u(x)$ [22]. In [36] Shenker gave the shape of utility function curves for both elastic and real-time traffic flows. Nowadays many papers focusing on utility function have been published. However, most of them were concentrated on the utility function of only one type of traffic flow and did not consider a more realistic scenario that multiple types of traffic flows are carried by a network. Some other papers which considered multiple types of applications set priority levels for different type of traffic flows and preempted traffic flows based on priority levels when necessary. Such priority level definition may not be flexible enough for today's and future Internet services.

As mentioned before, the initial work for utility function has been done in [36] and [22]. Utility functions curves of best effort traffic and real-time traffic are given in [36]. Best effort service is also called elastic applications in that paper. It includes traditional data applications which are tolerant of delays. Its curve appears to have decreasing marginal improvement due to incremental increases in bandwidth (see Figure 2.1(a)). The utility function curve of real-time service such as VoIP or video conferencing looks like a step function, user will gain a fixed utility when bandwidth requirement is



(a) Elastic application. (b) Real-time application.

Figure 2.1: Utility function curves.

satisfied. However today's real-time applications can be implemented to be tolerant of occasional delay-bound violations and dropped packets, so the utility function curve for a delay-adaptive real-time application looks like the one in Figure 2.1(b). In [22] the author also presented an economic framework for traffic management with similar utility function description. In this framework, the more utility is, the more satisfied the user feels.

The difference of utility function curves between elastic traffic and real-time traffic is that real-time traffic has some minimum "requirements". This minimum requirement can be on bandwidth, throughput, delay, or their combination, etc. If the minimum requirements could not be satisfied then users' utility would become zero or less. Although negative utility may be meaningful (i.e. how "dissatisfaction" the user feels), in this paper we set the utility equal to zero whenever the basic requirement can not be satisfied. Admission control will refuse to setup those connections whose utilities equal zero. We will not initial the transmissions which will make our users dissatisfied.

2.1.1 Functions for Elastic Application

Based on the shape of utility curve given by Shenker, utility function is used in many literatures focusing on resource allocation for elastic applications. In [20] and [21], Kelly uses a logarithmic utility function which is strictly concave as utility function for elastic applications as follows:

$$U_r(x_r) = w_r \log x_r$$

where x_r is the bandwidth taken by the elastic application on route r, and w_r is called "willingness to pay", which is defined as the amount to pay by user per unit time. By solving a constraint optimization problem using above utility function as follows:

$$\max \sum_{r \in R} w_r \log x_r$$

subject to

$$Ax \leqslant C$$

over

$$x \geqslant 0$$

where $Ax \leqslant C$ ensures that the resources taken by applications on each route can't exceed its capacity, the result will lead to a so-called "Proportional Fairness".

Proportional fairness is more reasonable from an economic point of view

than max-min fairness [17], which means maximizing the network use allocated to the session with the minimum allocation [7]. We use an example similar to the one in [33] to show this. The topology is shown in Figure 2.2.

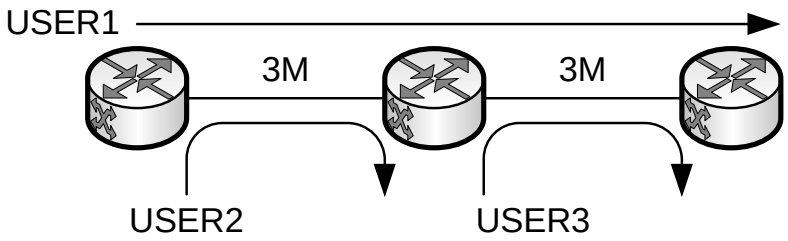


Figure 2.2: Network topology to show the difference between proportional fairness and max-min fairness.

In Figure 2.2 there are three applications: USER1, USER2 and USER3. In max-min fairness mode, each user should take a bandwidth of $1.5M$. However, USER1 takes two links, and from economic aspect, serving USER1 will cost more than serving the other two applications. If we define a logarithmic utility function as:

$$U(x_i) = \log(x_i)$$

where x_i is the bandwidth taken by USERi, and try to solve following constraint optimization problem:

$$\max(\log(x_1) + \log(x_2) + \log(x_3))$$

subject to

$$x_1 + x_2 \leqslant 3 \text{ and } x_1 + x_3 \leqslant 3$$

over

$$x_i \geqslant 0$$

The total (aggregate) utility here is $\log(x_1) + \log(x_2) + \log(x_3)$. Obviously if the total utility reaches its maximum value, the resources must be full utilized. We let $x_1 = x$, thus the total utility becomes $\log(x) + \log(3 - x) + \log(3 - x)$ where $0 \leqslant x \leqslant 3$. The function curve is shown in Figure 2.3. We

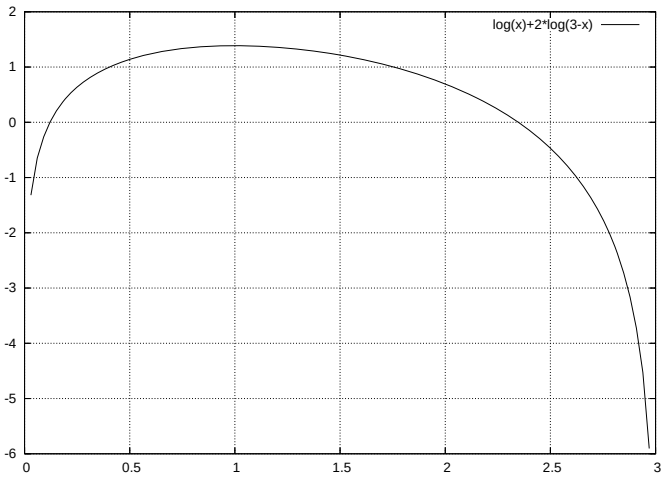


Figure 2.3: Total utility of three applications with logarithmic utility functions.

get the result that when $x_1 = 1M$, $x_2 = x_3 = 2M$, the total utility will reach its maximum. In this case, all of three applications take $2M$ resources. From an economic view, they are balanced. So comparing with max-min fairness, we believe the proportional fairness is an "economic" or cost fairness, while max-min fairness is a "service" fairness.

Following Kelly's research work, many literature such as [23], [32], [33], [10], [39] and [24] have addressed the bandwidth sharing problem of elastic applications especially TCP-like connections as an optimization problem for

logarithmic utility functions. Based on Kelly's research work Kar etc. propose a distributed algorithm in [19] to maximize total utility for elastic applications with logarithmic utility functions. [11] extended the utility function for TCP connections and hence proposed a utility function for HTTP-like connections. Steven Low did a deep research in utility function for TCP in [29]. He proposed utility functions for different TCP algorithms including TCP-Reno and TCP-Vegas.

In [24] Kunniyur and Srikant also mentioned a utility function for TCP as follows:

$$U(x) = \log \frac{x}{1+x}$$

2.1.2 Functions for Real-time Application

Unlike the research in using utility function for elastic application traffic engineering, we found few literature suggesting utility function for real-time applications. We believe this is because unlike the utility function curve for elastic application which is strictly concave everywhere, we can hardly find one simple function which is convex first then concave. Furthermore, from [28], [27] and [19], we know that total utility will have one unique maximum value under some constraints (capacity constraint) when utility functions are strictly concave. Even if we can find such utility function, it is difficult to find the maximum value of total utility. So we wish to still use a concave utility function for real-time application, although we know Shenker's curve for real-time applications is not always concave. Another conflict is that from [36] we know that for real-time application, admission

control is required. However, for any utility function that is strictly concave, the network must be no admission control to maximize the total utility. The ideal utility function we need for real-time application should both support admission control and be strictly concave.

In [13] a utility function is proposed as follows:

$$u(b) = \frac{1 - e^{-\beta(b-b_m)}}{1 - e^{-\beta(b_r-b_m)}} \tag{2.1}$$

where the b_r is desired bandwidth of a request and b_m is the minimum bandwidth.

Minimum bandwidth is the basic requirement, if b_m can not be satisfied the transmission will not be initialized. When b_m bandwidth is allocated we say the transmitting result is "acceptable" to users. Desired bandwidth b_r is the maximum bandwidth can be consumed by an application. If b_r is allocated for an application, its utility will reach a maximum value and will not increase even more bandwidth are allocated. At that time users feel fully satisfied for the transmission.

With the introduction of b_m we can solve the conflict between admission control and concave curve utility function. When available bandwidth is smaller than b_m the application will be rejected into network. And when bandwidth is greater than b_m the utility function curve is concave. Thus by running balancing algorithm a unique maximum total utility can be achieved.

Dharwadkar et al also set up utility functions in [13] for applications with fixed bandwidth requirement whose function are step functions, and for

applications whose utility functions are linear. The linear utility function is defined as:

$$u(b) = \frac{b - b_m}{b_r - b_m}$$

where the b_m and b_r are same as above. And for step utility function:

$$u(b) = \begin{cases} 1 & \text{if} \quad b \geq b_r \\ 0 & \text{if} \quad b < b_r \end{cases}$$

In [13] a preemption heuristic is designed based on above utility functions. Degrading algorithm for applications with linear or concave utility functions is also proposed. For applications with concave utility function the degradation is based on the result of "Gradient-Projection Method" which we will discuss in next section. For applications with linear utility function the degrading decision is based on the consuming and lacking amount of bandwidth.

2.1.3 Other Miscellaneous Functions

Another traffic engineering heuristic for MPLS networks is proposed in [14]. In this paper Elwalid et al use end-to-end delay as performance evaluation criteria, and try to minimize total delay. They achieve this objective by sending probe packets to get the end-to-end delay and its derivative on traffic load change, then using "Gradient-Projection" method to minimize total delay. The innovation of this heuristic is that Elwalid et al do not use any utility

function to describe the relationship between user's satisfaction degree and application consumed resources, but use probe packets to estimate the performance directly. [14] gives us the inspiration that it may not be necessary to use utility function to describe user's satisfaction degree.

A versatile preemption policy for MPLS networks was proposed by de Oliveira et al in [12]. They considered the some different objectives when preempt LSP(s). These objective include:

- minimize the priority of preempted LSP(s),
- minimize the number of preempted LSP(s),
- minimize the preempted bandwidth.

They define $\mathcal{L}$ as LSP(s) having a lower priority level than the LSP setup request, $b(l)$ is the bandwidth reserved by LSP $l \in \mathcal{L}$, $y(l)$ is the preempting cost (here it is the same as priority level). So for a preemption decision vector z which is defined as:

$$z(l) = \begin{cases} 1 & \text{if LSP } l \text{ is preempted;} \\ 0 & \text{otherwise} \end{cases}$$

a combination cost function is defined as:

$$F(z) = \alpha(z \cdot y^T) + \beta(z \cdot 1^T) + \gamma(z \cdot b^T)$$

where y and b are priority and bandwidth vectors, respectively. Thus $z \cdot y^T$

refers to preempted priority, $z \cdot 1^T$ is the number of preempted path and $z \cdot b^T$ is the amount of preempted bandwidth. α, β and γ are their weight. The third term on preempted bandwidth can be changed into $(z \cdot b^T)^2$ to penalize the choice of preempted LSP which may cause high bandwidth wastage. A heuristic was proposed to minimize total $F(z)$, which performs better than the standalone preemption policy in simulation result.

2.2 Gradient-Projection Method

Gradient-Projection [7] is a general method to solve constraint optimal routing problem. In a constraint optimal routing problem, normally each link in a network is associated with a monotonically increasing convex cost function whose variable is traffic load on this link. The object is to minimize total cost under some constraints. Note cost function is required to be convex for minimizing total cost, while for utility function it's inverse. Utility function needs to be concave for maximizing total value.

Since each link has a cost function with it, for a certain traffic flow, the cost of transferring on a path is the summary of cost of each link along this path. The more traffic load, the more cost, and vice versa. The idea of Gradient-Projection method to minimize total cost is shifting traffic load from one path to another path. When some traffic load is shifted from path I to another path II, since the cost function is monotonically increasing, the cost of path I is decreasing and cost for using path II is increasing. If the cost decreased from path I is more than the cost increased from path II, then

total cost of whole network is decreased. To shift traffic load appropriately we need the information of the derivative of each path. i.e., the change of cost when we increase or decrease a "very small" amount of load on that path. Then we always shift traffic load from other paths to the path with minimum derivative (so that the cost increased on this path is smaller than the cost decreased from other path), until derivative of every paths are almost same. At this time the total cost is minimized. Since cost function is convex, i.e., the more traffic load, the more derivative of cost function, when we shift traffic load we will always decrease the derivative on shifting out path and increase the derivative on moving in path, so we always can find a unique solution.

The gradient-projection method is also often used to maximize total utility when traffic flow is associated with a concave utility function. The idea is quite similar. Since we need to maximize total utility, when we shift traffic flow from one path to another path, we need the utility increased from moving in path larger than the utility decreased from shifting out path. So we get the derivative of utility function of each path, and move traffic load from other paths to the path with maximum derivative until the derivative of every paths are almost same. Since the utility function concave we can always find a unique solution.

2.3 Multiple path routing algorithms

When more than one path exist between same source-destination pair, path selection is required. Several multiple path routing algorithms were proposed in literature, including two classic algorithms: widest-shortest routing (WS) [2] and shortest-widest routing (SW) [38]. Widest-shortest first routing algorithm chooses a path with the minimum hop count among all feasible paths. If there are more than one path, the path with maximum available bandwidth is selected. For shortest-widest first routing, the path with maximum available bandwidth among all feasible paths is chosen. If there are more than one path, the path with the minimum hop count is selected.

The widest-shortest first routing considers hop count of a path. This may cause bottlenecks when all flows transfer along one shortest path. The shortest-widest first routing tries to balance the load but may increase the number of hops and take more resources. Viewing this an algorithm named shortest-dist first routing which considers both hop count and link bandwidth usage is proposed based on max-min fair share [17]. In this algorithm, the path "distance" is defined as:

$$dist(P, n) = \sum_{i=1}^{k} \frac{1}{r_i^n}$$

where $r_1,\ldots,r_k$ are the max-min fair share rates of links on the path P with k hops. Here $(\frac{1}{r})^n$ is the polynomial link costs. By adjusting n, we can cover the range between shortest ($n = 0$) and widest ($n \to \infty$) path algorithms. An interesting point is that $dist(P, 1)$ can be interpreted as the *bit transmission*

delay from the traffic source to destination when this connection get the rate r_i at hop i. Results show that for best-effort applications (Shenker call them "elastic" applications. In this thesis, the term "elastic" application and "best-effort" application are interchangeable) the Shortest-$dist(P, 1)$ path algorithm performs better than WS and SW in most cases by balancing the weight given to the "shortest" and "widest" metrics appropriately [31].

A systematic evaluation of WS, SW and shortest-dist algorithm was given in [30], where Ma and Steenkiste focused on routing algorithms for guaranteed applications. Their results show that when network load is heavy, algorithms tending to limit the hop count such as WS have lower bandwidth blocking rate than others giving preference to balancing the network load such as SW and shortest-dist. But when network load is light the results are on the contrary. Ma and Steenkiste believed that when network load is heavy congested links become unavailable for guaranteed sessions.

Viewing that conventional shortest-path routing often causes bottlenecks due to single path routing, in [6]and [5] Bak et al proposed a routing and scheduling scheme which is easy to implement. They tried to distribute the traffic load over all available paths to the destination for load balancing: load-balancing routing via full randomization (LBR-FR). To prevent routing on very long paths they proposed a variation to distribute traffic load only on a few eligible paths: load-balancing routing via bounded randomization (LBR-BR). A node within distance k is chosen as an intermediate node to route. If k is fixed as average distance to each node reachable from the source

as follows:

$$k = \frac{1}{n} \sum_{i=1}^{n} dist(s, d_i)$$

where d_i is a node in the network and s is the source node, the algorithm is named as LBR-BR1. Here $dist(s, d_i)$ means the distance (i.e., hop count) between node s and d_i, which is different with the definition used in [31]and [30]. If k is dynamically chosen according to the shortest path length as follows:

$$k = dist(s, d) * \frac{\max\left(dist(s, d_i)\right) - 1}{\max\left(dist(s, d_i)\right)}$$

where d_i is a node in the network, s is the source node and d is the destination node, the algorithm is named LBR-BR2. Results show that LBR-BR2 is better than LBR-BR1, LBR-FR and shortest path first in performance with respect to throughput, message loss, delay and link utilization at most time.

Chapter 3

Bandwidth Management on Single Path

3.1 Utility Function for Real-Time Applications

As we discussed in Section 2.1, equation (2.1) is a feasible utility function for real-time application. We noticed that in this utility function, when $b = b_m$, $u(b) = 0$. As we discussed before, utility is the degree of end user's satisfaction. Thus when $b = b_m$, the application is permitted into the network, but the user's satisfaction degree is still zero which is the same as the utility when the traffic is not initiated. This contradicts the actual situation. We believe that as long as an application is permitted in the network, the user should have some degree of satisfaction. Then it should have some basic

utility.

Considering the multiple service environments we designed a combined function for real time applications based on function (2.1):

$$u(x) = \begin{cases} 0 & x < m \\ U + \alpha \left[\frac{1-e^{-\beta(x-m)}}{1-e^{-\beta(r-m)}} \right] & m \leq x \leq r \\ U + \alpha & x > r \end{cases} \tag{3.1}$$

(In (3.1) and from now on, we use r, m to replace b_r, b_m respectively)

Here we add big U as the basic utility the user will get if application is permitted to transmit on network. α and β are positive numbers. α is used to adjust the maximum utility, and β describes how "adaptive" the application is. The larger the β is, the more "hard" the application is, and the curve will be more similar to step function. The curves are shown in Figure 3.1.

One reason that we add a basic utility U in our utility function has been discussed before: when application is permitted into network, the end user should feel the result is "acceptable" and has some basic satisfaction degree. Another reason is that basic utility can work as priority level when considering preemption. When resources are not enough to accommodate new application and then preemption is required, application with lower basic utility will be preempted to make room for new-coming application with higher basic utility. The more basic utility one application has, the higher its priority is.

We compare our utility function curve and Shenker's utility function

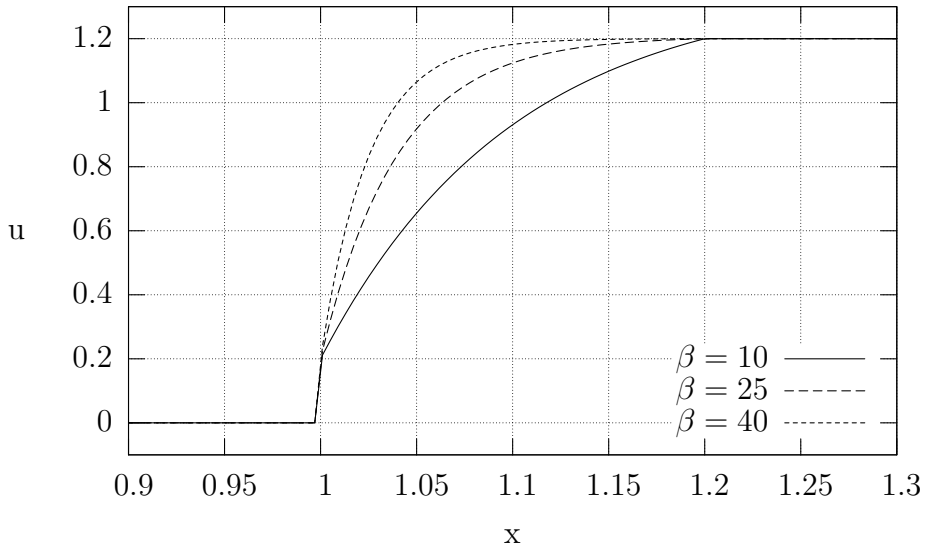


Figure 3.1: Utility Function Curves, m is 1, r is 1.2 and $U = 0.2$, $\alpha = 1$. We can tell when β becomes larger, the curve is more upright.

curve in Figure 3.2. Shenker's real-time utility function curve (Figure 2.1(b)) can be cut into two parts. The part before the point (m, U) is the convex part, in which the curve has a increasing marginal improvement. After the point (m, U) is the concave part, in which the curve has a decreasing marginal improvement. In our utility function we use a step function to replace the convex part in Shenker's real-time curve. Thus we can use the step function part to perform admission control, and with the concave part, we can use gradient-projection method to get a unique maximum total utility.

There comes a new problem when we try to maximize the total utility of a network. When two (or more) applications have same basic utility (priority) but with different bandwidth requirement, how can we maximize the total utility? For example, a network has a total bandwidth of $400K$ and has been already taken by one application with m at $400K$ and basic utility 10.

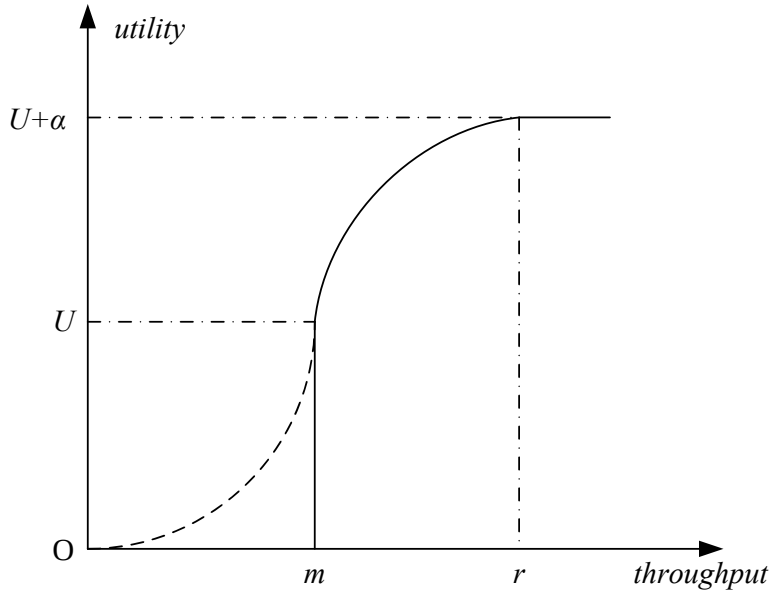


Figure 3.2: Comparison between our utility function curve (continuous line) and Shenker's utility function curve (dashed line). We use a step function to replace the convex part in Figure 2.1(b).

Now there come two new applications, whose m are both $200K$ and basic utilities are 10. We noticed that if the existed $400K$ application is replaced by two new $200K$ applications, we will increase the total utility from 10 to 20. However, since all three applications have same basic utilities, the existing $400K$ application can't be preempted by two new-coming $200K$ applications, and the two new applications will be rejected due to lack of resources. From this point of view, the total utility is not maximized. This is because our basic utility has nothing to do with the resource amount that the application has taken (reason will be explained in paragraph afterward). So the maximization of total utility by preemption in our heuristic is conditional. One application can only be preempted by another application with higher basic utility.

Considering the situation of unconditioned preemption, i.e., application

can be preempted as long as the preemption behavior can increase the total utility. In last example, the existed $400K$ application will be preempted by two new $200K$ applications. But if later there comes some applications with m at only $100K$, and still their basic utilities are 10, then the $200K$ applications will be preempted by $100K$ ones. Obviously, if unconditioned preemption is applied, applications with high resource requirement are biased. That's why we only use conditioned preemption in our heuristic.

The basic utility of an application is the end user's satisfaction degree when he/she feels that the result of transmission is just "acceptable". A VOD (Voice-On-Demand) user who takes a $400K$ bandwidth may not feel more satisfied than a VoIP (Voice over IP) user who consumes only $100K$. Also, from the aspect of priority level, application requiring more resource may not be more "important" (have higher priority level) than application taking small resource. So we decide that the basic utility of one application should have nothing to do with the resource taken by it. It only shows the "importance" of an application. For example, a VoIP meeting application for military use may have more basic utility than a online video chat application. From the economic point of view, an application taking more resource only means that its owner will be charged more. Serving two $200K$ applications can't earn more money than serving one $400K$ application, so there is no reason to preempt one $400K$ application with two $200K$ ones which have same basic utility. Preemption only occurs when new application is more "important" (has more basic utility) than other application. Otherwise, if all applications have same basic utilities, they are served in time order (FCFS).

To sum up, our utility function for real-time applications (equation (3.1)) includes two parts: basic utility and the concave curve part. Basic utility is used for admission control. Once a real-time traffic flow is permitted into the network, a bandwidth balancing algorithm using the remaining part of utility function (the concave curve part) will be executed to maximizing the total utility. In our heuristic which will be discussed in detail in next section, total utility maximization step is always performed after the admission control step.

Our utility function not only can solve preemption problem, but also support services with different bandwidth requirements, such as minimum rate guaranteed service, upper bounded rate service, and minimum rate guarantee upper bounded service [25].

For minimum rate guaranteed service, we let $r = \infty$, thus we got:

$$u(x) = \begin{cases} 0 & x < m \\ U + \alpha \left[1 - e^{-\beta(x-m)}\right] & x \geq m \end{cases} \tag{3.2}$$

For upper bounded rate service, we just set $m = 0$, and then we have:

$$u(x) = \begin{cases} 0 & x = m \\ U + \alpha \left[\frac{1-e^{-\beta * x}}{1-e^{-\beta * r}}\right] & 0 < x \leq r \\ U + \alpha & x > r \end{cases} \tag{3.3}$$

For minimum rate guarantee upper bounded service, equation (3.1) can support it well.

Sometimes some best-effort application users wish to pay for a "premium" service. For such service they also need utility functions. However, since they are best-effort application, they should have neither minimum rate guarantee nor rate upper bound. In such cases, we let $m = 0$ and $r = \infty$, and the function is as follows:

$$u(x) = U + \alpha(1 - e^{-\beta x}) \tag{3.4}$$

Notice that for utility function (3.3) and (3.4), since they have no minimum requirement, they can always be permitted into network, i.e., there is no admission control for such applications.

3.2 Heuristic for Bandwidth Allocation

In our scheme, QoS requirements for a real-time application are needed to be requested explicitly. QoS requirements information should include utility function parameters as α, β, basic utility U, minimum bandwidth m and requested bandwidth r. The QoS information should be either included in packets, or included in a signaling sent to MPLS ingress before LSP is set up. If MPLS network receives a packet without any QoS declaration, it will treat the packet as from a best-effort application.

As stated in previous section, our utility function is composed by a basic utility part (step function part) and a concave function part. The basic utility is used to perform admission control, then the concave part is used to

maximize total utility.

3.2.1 Admission Control

The flow chart of admission control is shown in Figure 3.3. Our admission control is composed of three major parts: process for new best-effort applications, process for new real-time applications and process for pending list. Applications are first divided into two categories: best effort applications and real-time applications.

Since best effort applications do not have any minimal requirement, we assume such applications are free to serve. So we didn't define any utility function for them. Also there is no admission requirements for best effort applications [36]. As we reviewed in Section 2.1, proportional fairness is an "economic" fairness, which means users can consume same amount of resources if they pay the same. However, since our best-effort applications are free to serve, we believe that the max-min fairness which is "service" fairness is more preferable. So we simply transmit all best-effort applications based on max-min fairness. (Process for new best-effort traffic part in Figure 3.3.) For those "premium" best-effort applications, we use the utility function (3.4) and treat them as real-time applications.

For new real-time application, we first let all all existing traffic transfer at their minimum bandwidth, then check whether the available bandwidth A is enough for new traffic's minimum bandwidth ($A > m_{new}$). If the available bandwidth is enough then the new real-time traffic is permitted into the

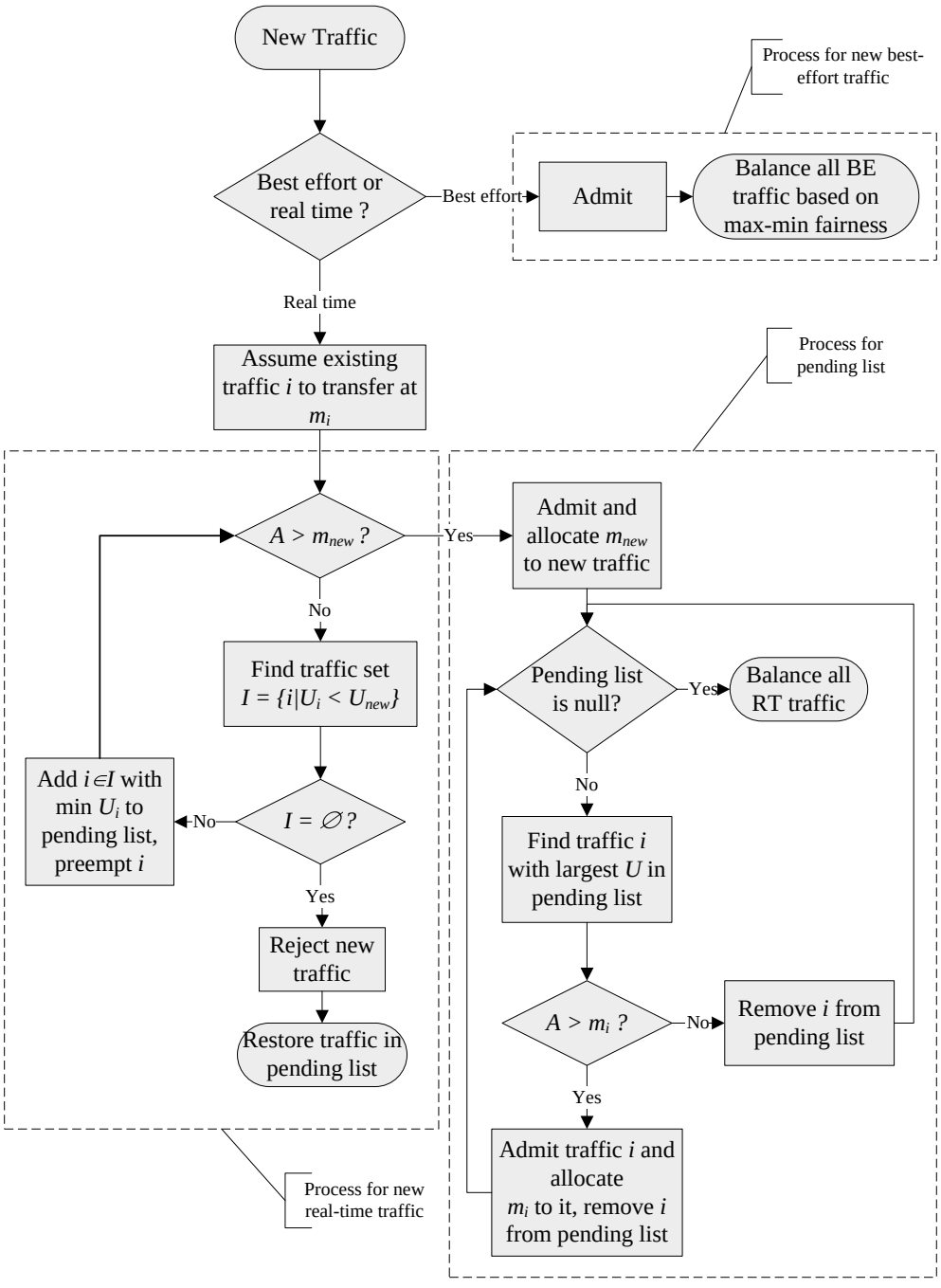


Figure 3.3: Flow chart of admission control for single path case.

network. After that balancing algorithm will be run to maximize total utility.

If the available bandwidth is still not enough for new application's minimum bandwidth requirement, that means either one or more existing traffic will be preempted, or the new traffic will be rejected due to lack of resources. We first find all traffic whose basic utility is smaller than the new traffic's, and put them in a set I. If $I = \varnothing$ that means no existing traffic has smaller basic utility, thus the new traffic will be rejected.

If I is not empty, we pick out the traffic with minimum basic utility, preempt this traffic and add it to a pending list. Then we re-calculate the available bandwidth and check if $A > m_{new}$. If yes then the new traffic is admitted. If the available bandwidth is still not enough then we update the set I and repeat the step of picking traffic with minimum basic utility, until available bandwidth is larger than new traffic's minimum bandwidth requirement, or the set I becomes empty then the new traffic will be rejected, and all traffic in pending list are restored. (Process for new real-time traffic part in Figure 3.3.)

After new real-time traffic is permitted, if the pending list is not null we will check whether traffic in pending list can be accepted in decreasing order of basic utility. By using pending list we can accept more traffic than simply preempt all traffic with lower basic utility. As a example, we assume we have a path with a capacity of $300K$. Flow 1 with $m_1 = 50K$ and $U_1 = 10$ and a flow 2 with $m_2 = 150K$ with basic utility $U_2 = 15$ are transferring on it. Now there comes a new traffic flow 3 with $m_3 = 200K$ and $U_3 = 20$. Since the available bandwidth is not enough for flow 3 and the existing two traffic

flows 1 and 2 have lower basic utilities than the flow 3's, flows 1 and 2 are preempted and added into pending list, flow 3 is accepted. However, after flow 3 is permitted the available bandwidth still can accommodate flow 1. So instead of simply preempting flow 1 and flow 2, we decide to let flow 1 in. For flow 2, though it has more basic utility than flow 1, it demands so much bandwidth that even if we preempt flow 1 the bandwidth is still not enough, so we reject flow 2 due to lack of resources. (Process for pending list part in Figure 3.3.)

3.2.2 Maximizing Path Utility

After admission control we need to reallocate bandwidth among traffic flows (load balancing). As stated before, for best-effort applications we simply balance them based-on max-min fairness.

For balancing among real-time applications, concave part in our utility function is used to maximize total utility of the path. The main idea we use is similar to Gradient-Projection algorithm. As mentioned in Section 2.2, Gradient-Projection method is used to minimize a total cost value, so it always divert traffic from paths with higher derivative of cost to paths with lower derivative. In our balancing heuristic, since we try to maximize the total user utility, traffic flows with lower derivative of utility will be degraded to give up their consumed bandwidth to those with higher derivative.

The derivative of our utility function (3.1) is:

$$u'(x) = \begin{cases} \alpha\beta\frac{e^{-\beta(x-m)}}{1-e^{-\beta(r-m)}} & m \leq x \leq r \\ 0 & x > r \end{cases} \tag{3.5}$$

The derivative function (3.5) is used in bandwidth balancing algorithm. As we discussed before the balancing algorithm is performed after the admission control step. Since all real-time applications that are permitted into the network must take their bandwidth larger than their m, we omit the $x < m$ part in equation (3.5).

The balancing algorithm is used to distribute available bandwidth among admitted traffic flows to maximize total utility. Following are its steps:

1. Get utility derivative of all traffic flows;
2. Increase reserved bandwidth of the traffic flow with largest derivative by a pre-defined step size s;
3. Re-calculate this flow's derivative;
4. Repeat from step 2 until all available bandwidth is allocated.

By using the balancing algorithm, the total utility can be maximized. Notice that we say the total utility is maximized only considering all existing applications; the resources are distributed among these applications to maximize their total utility. But for the case of new connection requirement, sometimes the total utility may be really decreased. For example, when a new connection request comes in, but the network can't accommodate it, some application must be degraded. Suppose the degraded application lost

utility u_1, while the basic utility of new connection is u_2. If u_1 is greater than u_2, when we accept the new connection requirement, the total utility actually becomes smaller. However, we intend to accept as many applications as we can and to reduce the blocking probability as small as it can reach, but not just to maximize the total utility value. That is, between the choice of making fewer people quite satisfied, making more people feel not very good but still acceptable, we prefer the latter. The basic utility is more like a priority level to decide preemptive behavior, while the utility function is used to make the allocation of resources more reasonable.

Note that the operation on changing bandwidth allocation is made all at once. In our heuristic and balancing algorithm, we sometimes need to *ASSUME* the traffic transfer at its m and then increase its resource step by step. This is only a calculation step. Application will not be degraded to its m first then be increased again. When we get the result of appropriate allocation we will shift the resource allocation at once.

3.3 Simulation Study

3.3.1 Simulation Scenario

We simulate a simple network with single path to demonstrate our admission control and bandwidth allocation procedures described in this chapter. The simulation topology is shown in Figure 3.4. In this figure, we have five pairs of source-destination nodes: node 0 is connected to node 7, node 1 is connected

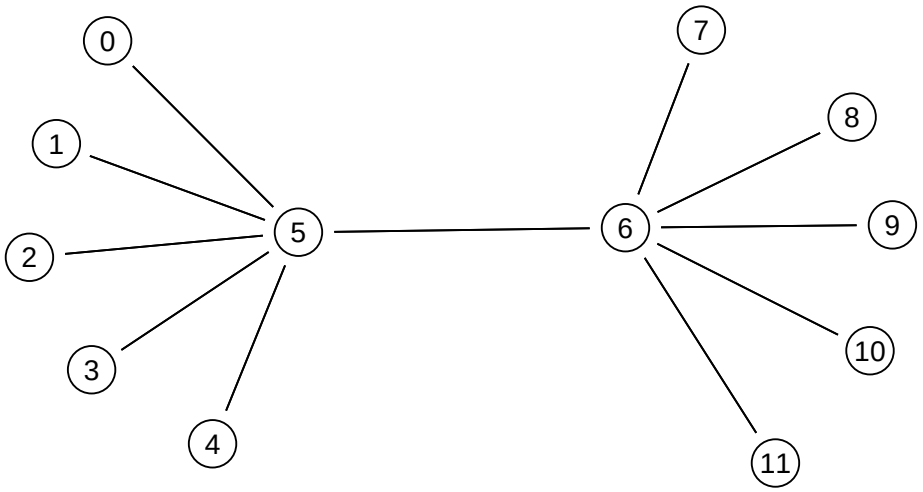


Figure 3.4: Simulation topology for single path case.

to node 8, node 2 is connected to node 9, node 3 is connected to node 10, and node 4 is connected to node 11. Node 0–4 are source hosts, node 7–11 are destination hosts. Node 5, node 6 and the single link between them form an MPLS domain, in which node 5 is the ingress, where our heuristic is applied. All links in this topology has a capacity of $500K$. For link in MPLS domain (i.e., link 5–6), to prevent starvation problem, we define that real-time applications can only consume no more than 90% bandwidth, i.e., $450K$. Otherwise real-time applications may consume up all bandwidth and cause the drop of all other packets including LDP packets which are used to set up or release LSPs.

There are 5 traffic flows in this network. Two best-effort traffic flows and three real-time traffic flows with different QoS requirements and basic utility. The details of them are shown in Table 3.1.

The S–D pair in Table 3.1 stands for source – destination pair, BE stands

Table 3.1: The details of traffic flows in coming order.

Flow Name	S–D pair	Type	Speed	α	β	m	r	U
LBE	0–7	BE	$50K$	N/A	N/A	N/A	N/A	N/A
HBE	1–8	BE	$100K$	N/A	N/A	N/A	N/A	N/A
LRT	2–9	RT	$150K$	1	0.1	$100K$	$150K$	10
HRT	3–10	RT	$500K$	1	0.025	$300K$	$500K$	15
PRI	4–11	RT	$400K$	1	0.053	$250K$	$400K$	20

for best-effort and RT stands for real-time. Packets in a BE flow are modeled as an exponential ON/OFF process with average ON and OFF time are all $0.5s$, and the mean rate at ON time is shown in *Speed* field. Packets in RT flows are modeled by a Poisson process with a mean arrival rate defined in *Speed* field of the table.

Traffic flows come into network in an order described in Table 3.1. The time interval between each two consecutive traffic flows is $5s$, and the total simulation time is $30s$.

We set up our simulation using Network Simulator 2 [37] with the help of a third-party MPLS simulation package: MNS [1].

3.3.2 Result Analysis

The throughput of the 5 traffic flows during simulation time is shown in Figure 3.5. LBE comes first after simulation started. Since LBE is best-effort traffic flow, it is permitted into network with no bandwidth reservation. And because the network is empty, LBE is transferred at full rate. At $5.0s$

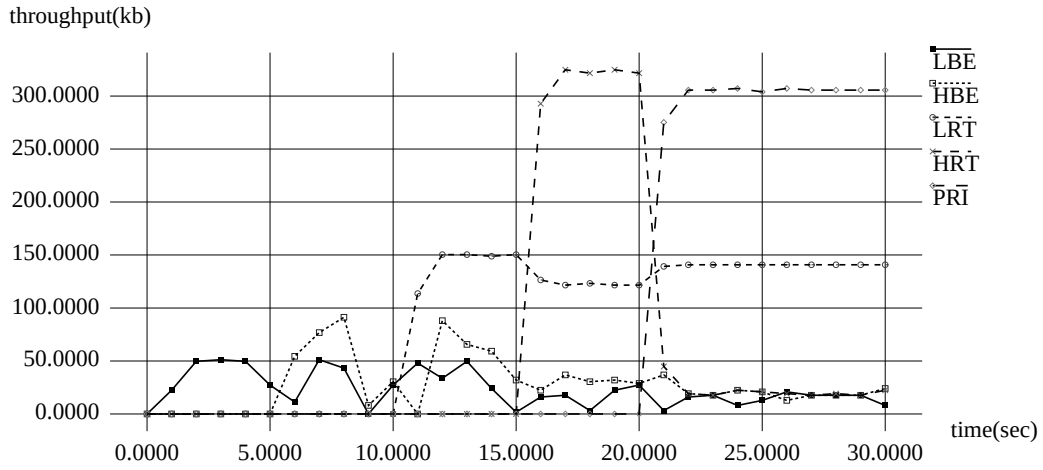


Figure 3.5: Throughput of traffic flows in single path case.

HBE comes, like LBE it is also permitted in with no reservation. Since the resource was still enough it is also transferred at its arrival rate. LRT comes at $10.0s$, it is a real-time application and has claimed its requested bandwidth $r_{LRT} = 150K$ and minimum bandwidth $m_{LRT} = 100K$. The ingress (Node 5) checks available bandwidth $A = 450K$ and finds it is larger than m_{LRT}, so it accepts LRT and reserves bandwidth for it. Because $A > r_{LRT}$ the network reserves $150K$ bandwidth for LRT's LSP.

Things become complex when HRT comes at $15.0s$. HRT claims its requested bandwidth $r_{HRT} = 500K$ and minimum bandwidth $m_{HRT} = 300K$. According to the admission control decision described in Figure 3.3, we first assume all existing real-time traffic flows only consumed the bandwidth at their m. So we assume LRT took $100K$, and thus we have $A = 350K$, the available bandwidth is larger than m_{HRT}, HRT can be accepted. After that by running the balancing algorithm described in Section 3.2.2 with a step size of $1k$, we find that when LRT takes $121K$ and HRT takes $329K$ the

total utility is maximized. So Node 5 first degrades the bandwidth reserved for LRT's LSP from $150K$ to $121K$, then sets up a LSP with $329K$ reserved for HRT. From Figure 3.5 we also notice when HRT and LRT consume up $450K$, LBE and HBE share the remaining $50K$ based on max-min fairness.

When PRI comes at $20.0s$, again we assume that LRT and HRT take their minimum bandwidth. We get $A = 450K - 100K - 300K = 50K$. Since $A < m_{PRI}$, either LRT and/or HRT will be preempted, or PRI is rejected. According to our admission control heuristic, we first found set $I = \{HRT, LRT\}$. Since LRT has a lowest U, we assume to preempt (we use the word "assume" to show that we do not really preempt LRT now, it's just a calculation step) LRT and added it into pending list. Then we update $A = 150K$, and find that it is sill not enough for accepting PRI, so again HRT was assumed to be preempted and added into pending list. Now $A = 450K$ is enough for PRI, and PRI is accepted. After PRI is permitted into network we find that HRT and LRT are in pending list. We first check HRT and find it can not be accepted any more, this time it is really preempted. For LRT we find that we still can accept it. So though we have $U_{LRT} < U_{HRT}$, HRT is preempted due to lack of resource, and LRT is still acceptable. By running balancing algorithm we see that when LRT takes $139K$ and PRI takes $311K$ the total utility would be maximized.

The real operations on Node 5 at $20.0s$ are:

1. the LSP for HRT is released, no bandwidth reservation for HRT from now;

2. setup a LSP with $311K$ bandwidth reserved for PRI;

3. increase bandwidth reservation for LRT's LSP from $121K$ to $139K$.

We found from Figure 3.5 after $20.0s$ LRT and PRI consumed their allocated bandwidth, HRT, as well as LBE and HBE, share the remaining $50K$ based on max-min fairness.

The change of total utility is drawn in Figure 3.6. The total utility became more than zero when the first real-time application came at $10.0s$. After that each time there came a new real-time application, the total utility increased.

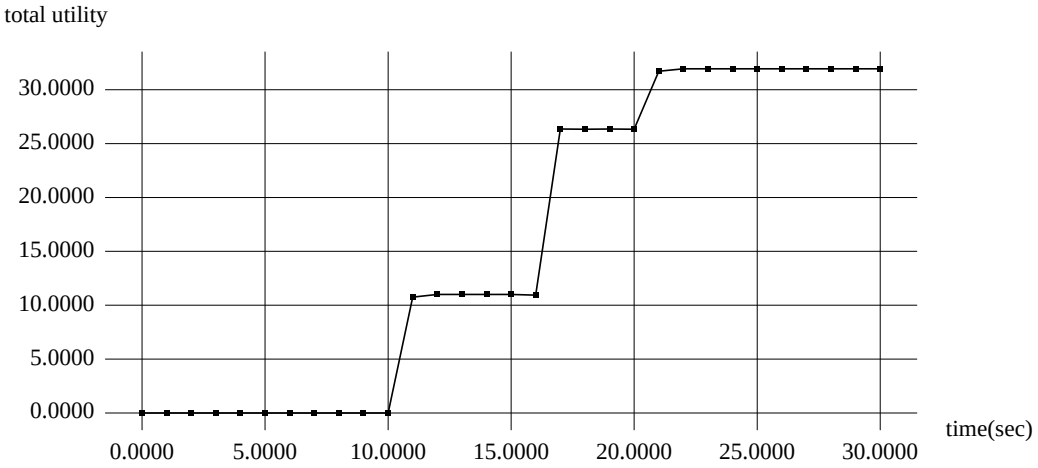


Figure 3.6: Total utility vs. time in single path case.

Since the basic utility U is a significant part in our utility function, we re-draw the total utility without basic utility part in Figure 3.7 to study the result of our balancing algorithm. We can see between $10.0s$ and $15.0s$ there was only one real-time application in our network, so without considering basic utility part, the total utility is at its maximum value: 1. After $15.0s$

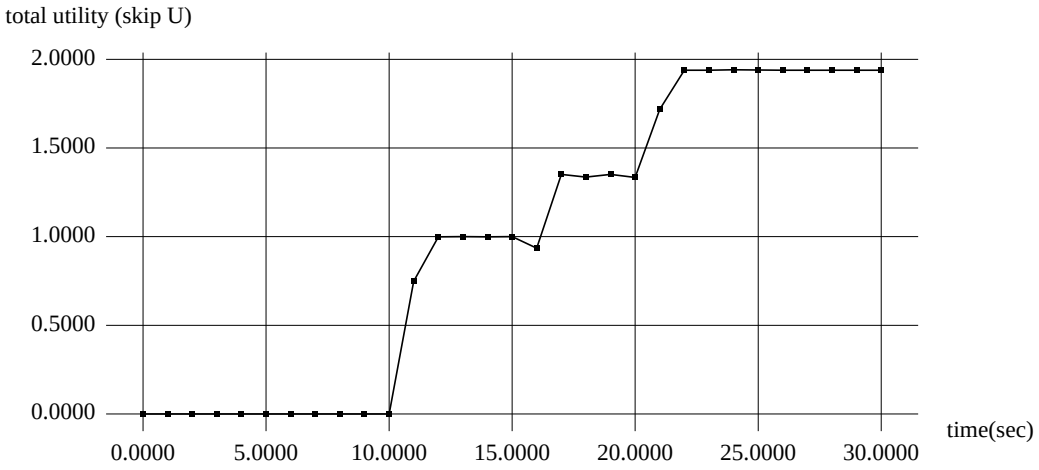


Figure 3.7: Total utility (without basic utility part U) vs. time in single path case.

there always be two real-time applications with $\alpha = 1.0$, and because network resource is not sufficient for them to transfer at full rate, without basic utility part the total utility should smaller than 2. We noticed that when HRT comes at $15.0s$ there is a drop in total utility. This is due to we degraded LRT before HRT's LSP was set up.

3.4 Summary

In this chapter we first proposed a type of utility functions for real-time applications with different QoS requirements. The utility function contains a basic utility indicating the minimum utility user can get when his/her transmission request is permitted, and a function whose curve is strictly concave indicating the change of user's satisfaction degree according to the change of its consumed bandwidth. Then the heuristic for bandwidth management in a

multiple service class environment on single path is described. In our heuristic, best-effort applications are always accepted for no charge and transferred based on max-min fairness. For real-time applications, our proposed utility functions are divided into two parts. The first part is basic utility, which is used for admission control. Preemption decision is also made based on it. The second part forms a concave curve, which is used to balance resources among existing applications to maximize total utility. Finally we use network simulator to demonstrate our heuristic by showing bandwidth allocation result. The change of total utility is also presented.

Chapter 4

Bandwidth Management on Multiple Path

4.1 Maximizing Total Utility

The utility functions we proposed in Chapter 3 can also be applied in a multiple path network. However, comparing to the admission control and resource allocation on single path, when we try to maximize total utility in a network with multiple paths like the topology used in [13], we come to the path selection problem, for both best-effort application and real-time applications.

Our maximizing total utility heuristic for multiple paths is shown in Figure 4.1. Like we suggested in the single path case, when ingress LSR receives a new transmission request, it first classifies into best-effort application or

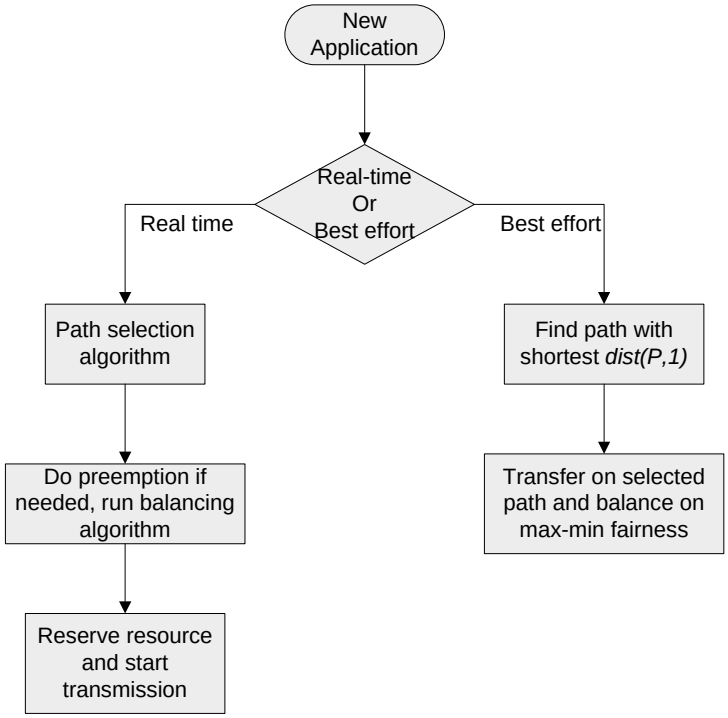


Figure 4.1: Heuristic for multiple path.

real-time application. Then for both best-effort applications and real-time applications the heuristic contains a path selection part and a balancing part. The path selection part will choose an appropriate path from all available paths or reject the new request if resources are not enough. The balancing part will only be executed on selected path after request is accepted.

4.1.1 Path Selection for Best-effort Applications

For best-effort application, as we reviewed in Section 2.3, the $dist(P, 1)$-first routing proposed in [31] performs better than other multiple path routing algorithms. So we use this algorithm as our routing algorithm for best-effort

applications. When a best-effort application requires enter network, ingress LSR calculate:

$$dist(P, 1) = \sum_{i=1}^{k} \frac{1}{r_i}$$

on each available path for it. Note that here r_i is the max-min fair share among all best-effort applications (i.e., the *remaining* resources after reserving for real-time applications), not among all applications on that link, as only best-effort applications are balanced according to max-min fairness. Then ingress LSR will choose the path with minimum $dist(P, 1)$ to transfer this application. There is no admission control for best-effort applications, all best-effort requests will be granted into network.

4.1.2 Path Selection for Real-time Applications

Unlike best-effort applications, real-time applications normally have some minimum bandwidth requirements and need admission control. So the path selection for real-time applications are much more complex. In [9] and [8], we noticed that in optical network the optical wavelength assignment algorithm has impact on network performance such as total throughput and blocking probability. We also notice the existing multiple path routing algorithms that we have discussed in Section 2.3 are similar to wavelength assignment algorithms in optical networks. The shortest-path first is like the first-fit wavelength assignment, and the widest-path first is like the least-used first wavelength assignment. These give us the idea to combine different multiple path routing algorithms with our proposed utility function.

The multiple path routing algorithms we choose to use are WS [2], (SW) [38], and LBR-FR [6] (which we will call "RD" in the following). We also propose an algorithms totally based on our utility function, which we call "max-util" in the following contents. These four algorithms will be discussed separately in the following contents.

WS

In this algorithm we first sort all possible paths according to their lengths with shortest path first. If there are more than one paths with same lengths we sort them according to their available bandwidth, with widest bandwidth path first. Here the available bandwidth means the total bandwidth that can be consumed by real-time applications minus the sum of *minimum* bandwidth requirements of all present real-time applications.

After paths sorting we check the first path (the widest one) to see if the path can accommodate new application without preempting existing applications (i.e., whether available bandwidth is greater than m of the new request). If it can accommodate the new request, then the connection will be set up on this path. If there is no path can accommodate the new application then some existing path(s) will be preempted. Otherwise the connection will be set up on chosen path.

For preemption we need to check possible paths again according to widest-shortest first sorting. The process is the same as we proposed in Figure 3.3. We check if the path can accommodate the new application after all applications with smaller U are (assumed to be) preempted, if yes

the balancing algorithm will be run on this path, otherwise we check next path. After checking all possible paths if the new application still can not be accepted, it will be denied due to lack of resources.

Note the preemption operation is also "assumed" here, since we only need find an appropriate path. Real preemption operation will be executed in resource balancing operation which we will discuss later after path selection.

WS is the simplest algorithm among all of four algorithms.

SW

In this algorithm the multiple paths will be sorted according to available bandwidth (same definition as above). If more than one path have same available bandwidth then the shortest one comes first. Then we check the (shortest, if available) widest path to see whether its available bandwidth is greater than the m of the new request. If yes then this path is chosen to transfer, otherwise some transmission will be preempted. Here we only need to check the widest path since all other path(s) must have no ability to accommodate new request if the widest path can not accept it.

If preemption is required then we do the same operation as described in "WS", but this time the checking order of paths is according to available bandwidth.

This algorithm is a bit more complex than WS because it need to calculate available bandwidth on each path.

RD

Like "WS" algorithm we sort all the paths in randomly order, then check the them one by one to see whether preemption is needed. If preemption is needed we check the paths again to find the path can accommodate it after preemption.

Each time there comes a new transmission request a random algorithm is run to generate the order of paths. So for requests between same source-destination pair and have same multiple paths, in this algorithm their path order may not be same.

Compared with WS in the aspect of complexity, this algorithm need an additional step to shuffle the order of paths first.

Max-Util

This algorithm is totally based on our proposed utility functions. The idea is: since our aim is to maximize the total utility, so when a new request is permitted into our network, we choose the path that when the new request is transferred on it, the total utility will increase most, or decrease least in the case of utility decreasing (see 3.2.2 for explanation). If more than one path have this property, we choose the shortest one.

We first sort the path according to widest-shortest first order. Then for each path we calculate its current total utility. After that we try to accommodate new application on it by running the heuristic for single path case, and calculate the total utility after heuristic if the new application can

be accepted in this path no matter whether preemption is required. Then the path with maximal utility increment (or minimal utility decrement) is chosen. If no path can accept the new request, it will be rejected due to lack of resources.

Due to the complexity of the single path heuristic we proposed in Chapter 3, this algorithm is much more complex than the other three.

4.1.3 Resource Allocation After Path Selection

Resource allocation step is executed after path selection. For best-effort applications, as mentioned before, it will be transferred on selected path according to max-min fairness.

For real-time applications, the algorithm we run on selected path is the same as we described in Section 3.2.2. Also the preemption operation will be performed in this step.

4.2 Simulation Study

4.2.1 Simulation Scenario

The performance of proposed heuristic, including blocking probability and total utility of whole network, is evaluated by simulation. Both no preemption case and preemption case are evaluated. Since we tend to evaluate the performance of different multiple path routing and bandwidth allocation,

only real-time traffic flows which require admission control and bandwidth reservation are considered here.

The network topology we used in our simulation environment is a simple multiple paths topology. Five paths exist between one source–destination pair, named from Path I to Path V according to increasing order of the length of each path. Each path has the same capacity of $1M$.

Traffic flows in network are generated as an exponential on-off process. The average rate is r during the "ON" period, and is m as a whole. The packet size is fixed with 500 bytes per packet. We run two groups of simulations: without and with preemption. For the case of without preemption, all traffic flows have the same basic utility U. For the case of with preemption, traffic flows have four different basic utility level. The parameters are listed in Table 4.1.

Table 4.1: Random parameters of utility function used in generating traffic flows

Parameter	Value
m	randomly choose from $70K$ to $370K$
r	$r = m * p$ where p choose from 1.5 to 2.0
U (no preemption)	2
U (preemption)	choose from $\{1, 2, 3, 4\}$
α	1.0
β	$5.0/(r - m)$

We run a series of simulations starting with total number of applications equal to 5, and then 10, 15, . . . , until 70 in both without and with preemption cases. For each number of requests we run our simulation 100 times with

different random seeds selection to prevent from generating same parameters in each simulation (Random numbers generated by computer program is called pseudo-random. Same numbers are generated if same random seed is used). Average blocking probability and total utility versus number of requests of different multiple paths routing algorithms are compared. For the case of no preemption, their total utility with basic utility part skipped are also compared.

4.2.2 Result Analysis

In our simulation, minimal bandwidth requirement m distributed from $70K$ to $370K$ uniformly, the average minimum bandwidth for all traffic flows is $220K$. Since the capacity of each path is $1M$, the rejection of traffic flow requests is supposed to occur when we increase the number of requests to 25. We define the blocking probability as (number of rejected requests)/(number of total requests). For no preemption case, the blocking probabilities for our maximize utility heuristic, shortest-widest, widest-shortest and random routing algorithm are compared in Figure 4.2.

From Figure 4.2 we can see the rejection of requests occurs when we increase requests number to 20. However, the blocking probability is below 5%, that is, no request or only one request is likely to be rejected. Blocking occurs mainly when the number of requests goes higher than 25. In this case our proposed heuristic has the lowest blocking probability and shortest-widest first (the difference between them is slightly), then random. Widest-shortest has a highest blocking probability.

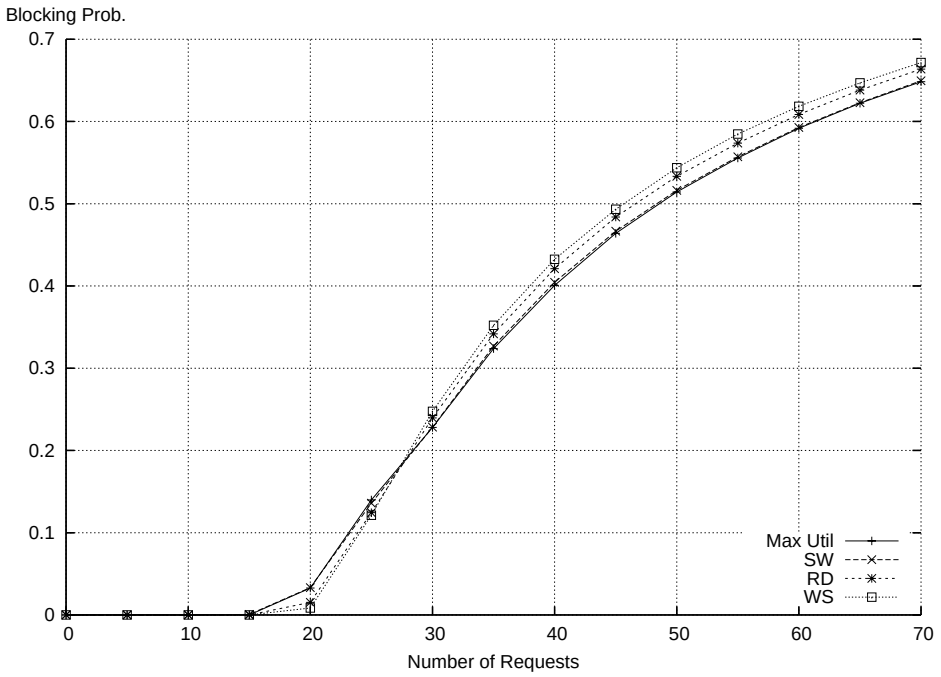


Figure 4.2: Blocking probability comparison for different multiple paths routing algorithms, no preemption case.

However when bandwidth is enough and there is no congestion (i.e., no preemption is needed), the result is opposite. WS has lowest blocking probability, then RD and SW, Max Util has highest blocking probability. We believe this is because WS always try to "pack" applications together since it always choose path with smallest length, and hence make room to accommodate more applications.

The total utility result of different routing algorithm are compared in Figure 4.3. We find in this figure all curves almost become horizontal when request numbers is larger than 25. This is because our network become "full" when request numbers is around 25, and since no preemption occur, those existing traffic flows will not be preempted and almost all subsequent requests

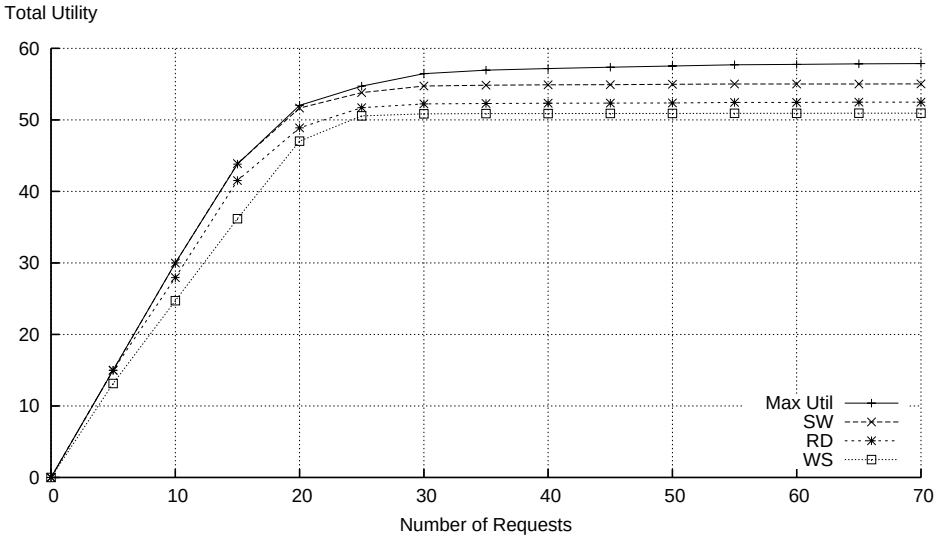


Figure 4.3: Total utility comparison for different multiple paths routing algorithms, no preemption case.

would be rejected. In all cases of requests number our proposed heuristic can achieve the highest total utility, then shortest-widest first, followed by random, widest-shortest has the lowest performance.

When there are 20 requests we notice that the Max Util has a higher blocking probability than the other three. We believe this is due to the parameter setting. Since our basic utility U is not significantly larger than α, as discussed in 3.2.2, sometimes accepting more requests may lower the total utility, so when our algorithm try to maximize the total utility, it blocks some requests. This will only occur when basic utility U is close to α, if blocking probability is important to users, they can simply increase the basic utility U to make U dominate the utility function.

We notice that when request number is below 25 where the network is not congested, although WS has lowest blocking probability, it has smallest

total utility. We believe this is because "pack" applications together means more applications need to be degraded.

The total utility result without basic utility part are compared in Figure 4.4. In this figure, all curves increase first, then drop and become flat.

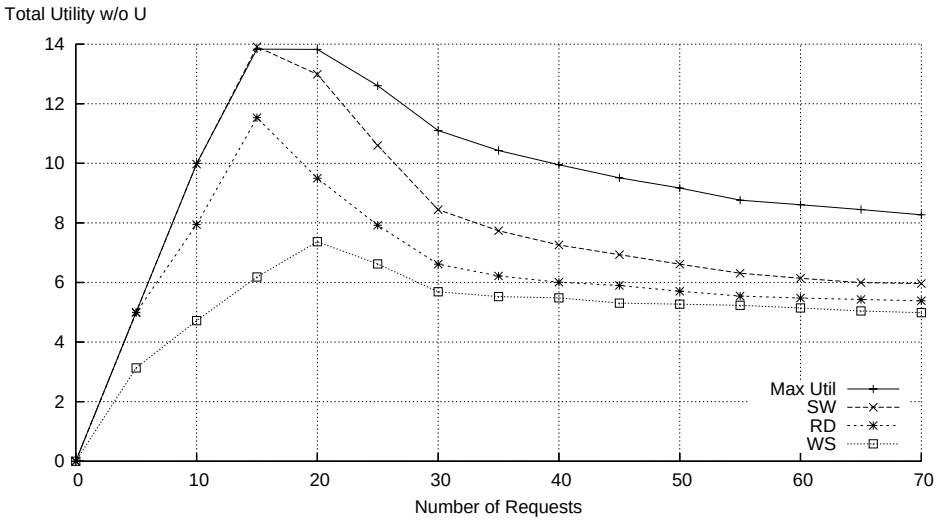


Figure 4.4: Total utility comparison (with basic utility part skipped) for different multiple paths routing algorithms, no preemption case.

This is because when requests number is low, traffic flows can take bandwidth at their requested bandwidth r, at this time the total utility without U increases fast. However, when more requests come existing traffic flows have to be degraded to accept new requests. Thus the total utility without U drops. Since there is no preemption, as discussed before, when network become "full", and almost all new requests are rejected, the curve of total utility without U becomes flat. For this performance, the order of four different algorithms is: max util heuristic > shortest widest first > random > widest shortest first.

The blocking probability comparison in preemption case is shown in Figure 4.5. Like Figure 4.2, when network is not busy, WS has lowest blocking

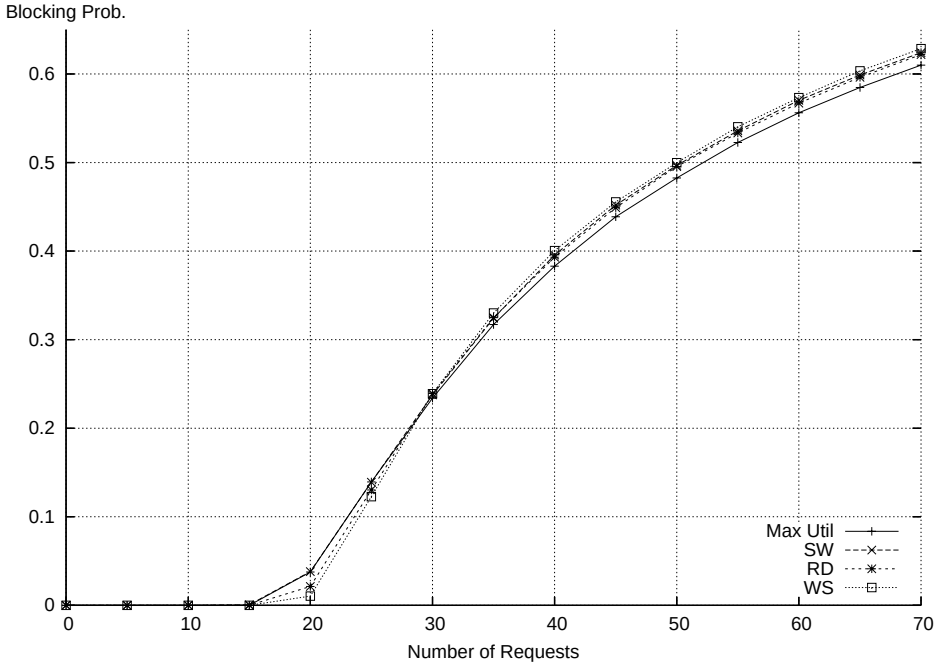


Figure 4.5: Blocking probability comparison for different multiple paths routing algorithms, preemption case.

probability, then followed by RD, SW, and Max Util has highest blocking probability. The reason is same as we discussed before. When the number of requests is larger than 25, the order of blocking probability of the four different algorithms is: max util heuristic < shortest widest first < random < widest shortest first. We noticed that in preemption case the blocking probability difference among shortest-widest first, random and widest-shortest first is very slight. This is because when the network is "full", their available bandwidth A (as we defined in Section 3.2.1) of all paths are almost same, in such case there is not much advantage to use shortest-widest first algorithm.

However, our max util heuristic can achieve lower blocking probability than the other three.

The total utility change of different algorithms are compared in Figure 4.6. The order of total utility of different algorithms is the same as in

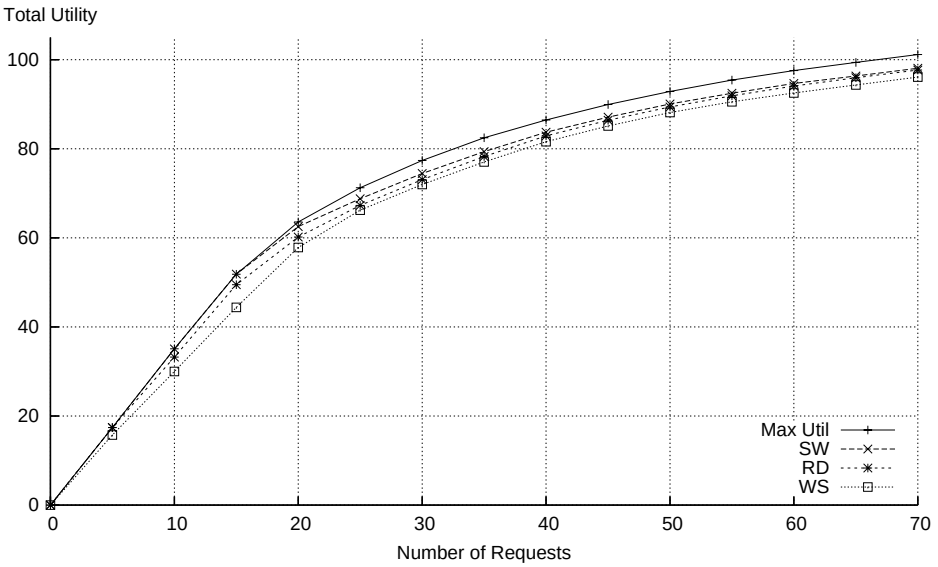


Figure 4.6: Total utility comparison for different multiple paths routing algorithms, preemption case.

no preemption case: max util heuristic > shortest widest first > random > widest shortest first. However, as we just discussed, when the number of requests becomes large the difference among shortest-widest first, random and widest-shortest first is not large. Max util heuristic still get the best result.

From the simulation results we have the following observation on how to choose path selection algorithm. If the objective is simply to maximize total utility, then our Max Util algorithm is the best. If the complexity of Max Util algorithm is not desired, as a compromise, SW is also a good choice.

If lower blocking probability is more important, we summary our result in aspect of blocking probability in Table 4.2:

Table 4.2: Blocking probability comparison among WS, RD, SW and Max Util

Traffic Load	WS	RD	SW	Max Util
Light	Best	Good	Good	Bad
Heavy (No Preemption)	Worst	Medium	Best	Best
Heavy (Preemption)	Medium	Medium	Medium	Best

Note that the terms such as "good" and "bad" in the table is relative when compared with other algorithms. So if the traffic load is light and no congestion in network, we suggest to use WS for its lowest blocking probability and simplicity. When traffic load is heavy and congestion occurs, if there is no consideration of preemption or all traffic flows are of same priority, both SW and Max Util are good choices, however, due to the complexity of Max Util, we recommend SW. When priority is considered and preemption is needed, Max Util is the best as long as complexity is not a fatal factor.

4.3 Summary

In this chapter, we have proposed a path selection algorithm, Max Util, based on our utility function introduced in Section 3.1. In this heuristic when a new traffic flow comes we test all paths to find the path that the total utility will increase most after the new flow is transferred on it. We have also evaluated the performance including blocking probability and total utility for

our proposed heuristic in multiple paths network. We have compared four routing algorithms: three existing ones (WS, SW, RD) and our Max Util. Both of the preemption case and without preemption case are considered. Simulation study shows that Max Util can achieve higher total utility in any condition and lower blocking probability when network becomes congested. Finally we made a suggestion on the choice of path selection algorithm under different condition.

Chapter 5

Conclusions and Further Research

5.1 Conclusion

Along with the development of multimedia and Internet technology, more and more applications with different QoS requirements arise and hence traffic engineering becomes more and more important. In view of this, in this thesis we have designed a bandwidth management scheme based on MPLS. The aim is to use one common network to meet the QoS requirements of today's and future applications as much as we can, and serve our end-user(s) as best as we can.

MPLS has many advantages in supporting traffic engineering. With the help of explicit routing supported by MPLS, it makes possible to force

some traffic flow(s) follow the route we designate. And with CR-LDP we can reserve certain amount of bandwidth for one traffic flow. Therefore, MPLS is ideal for implementing constraint-based routing for today's Internet.

After a brief introduction to Internet traffic engineering and MPLS architecture, we have reviewed the existing utility functions and their usage for both elastic and real-time applications. Then we have introduced the Gradient-Projection algorithm—a general algorithm suitable for solving constraint optimization problem. Finally several multiple path routing algorithms and the comparison of their performance have been introduced.

Maximizing end-users' total satisfaction degree is the final aim for all networking optimization methods. So in Chapter 3 we have first proposed a type of utility function for real-time applications which accords with Shenker's utility function curve. Then we designed a heuristic using our utility function to allocate and manage bandwidth for different service classes on single path. For best-effort applications we simply need do balance according to max-min fairness sharing. For real-time applications, the utility function is divided into two parts. The first part is used for admission control and the second part is used to balance traffic load. Degradation and preemption have been considered in our heuristic. Simulation results have been shown finally to demonstrate the heuristic.

In Chapter 4 we have considered the network with multiple paths. The proposed heuristic contains path selection part and balancing part. For best-effort applications, the shortest-$dist(P, 1)$ first path is chosen, and the share of resources among best-effort applications is according to max-min fairness.

For real-time applications, four different path selection algorithms have been proposed. Three of them are based on existing multiple path routing algorithms, and the other is based on the change of total utility. Then we have compared their blocking probability and total utility under different network conditions by simulation. The results have shown that Max Util has best performance when network is busy. Finally a suggestion on the choice of path selection algorithms under different network conditions has been made based on our simulation result.

5.2 Further Research

To apply the utility function and the heuristic in a practical network, we still have some future work to do. First and the most important is to determine the parameters used in the utility function. Few end-users will know the exact bandwidth requirements for their applications. Normally they only have requirements on delay, delay jitter, packet loss ratio, or even more obscure. This requires translation from these requirements into bandwidth requirement, and other parameters such as α, β and U appropriately. The translation job can be done by either running experiments under different circumstance to get statistical result from gathered result data, or doing mathematical analysis based on buffer management and packet scheduling algorithms. Moreover, as Shenker discussed in [36], all of these should be *explicitly* requested by applications, so some protocol need to be developed for the negotiating on utility function between end application and network

provider.

Another relating work is to find multiple paths effectively in a mesh network. The prerequisite for our heuristic is the existence of multiple paths, so finding the multiple paths effectively and accurately is important.

Re-routing is another helpful technique that can be considered. Sometimes with re-routing we can accept more traffic flows than without it. Several re-routing or alternate path routing algorithms in MPLS network such as [26], [16], [35] and [15] are proposed and performance such as total throughput, packet loss and blocking probability were improved. It is also extremely useful for route backup when some links suddenly fail.

List of Publications

1. L. N. Binh, N. M. Hoang, W. Zheng, A. Sekercioglu, "Modeling of wavelength routing and assignment in GMPλS-based DWDM optical networks", Conference on the Optical Internet & Australian Conference on Optical Fibre Technology, Melbourne, Australia, July 13–16, 2003.

2. L. N. Binh, N. M. Hoang, W. Zheng, C. Cieutat, A. Sekercioglu, "Routing and wavelength assignment in GMPLS-based optical networks", SPIE Optical Transmission Systems and Equipment for WDM Networking II, Orlando, Florida, USA, 7–11 September 2003.

3. Zheng Wu, Qinghe Yin, "A Heuristic for Bandwidth Allocation and Management to Maximize User Satisfaction Degree on Multiple MPLS Paths" accepted by IEEE CCNC 2006

Bibliography

[1] G. Ahn and W. Chun. Design and Implementation of MPLS Network Simulator (MNS). [Online]. Available: http://flower.ce.cnu.ac.kr/~fog1/mns/mns2.0/doc/MNS_v2.0_arch.pdf

[2] G. Apostolopoulos, S. Kama, D. Williams, R. Guerin, A. Orda, and T. Przygienda, "RFC 2676: QoS Routing Mechanisms and OSPF Extensions," Aug. 1999.

[3] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "RFC 3272: Overview and Principles of Internet Traffic Engineering," May 2002.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "RFC 2702: Requirements for Traffic Engineering Over MPLS," Sept. 1999.

[5] S. Bak, A. M. K. Cheng, J. A. Cobb, and E. L. Leiss, "Load-balanced routing and scheduling for real-time traffic in packet-switch networks," in *Proceedings of Local Computer Networks (LCN 2000)*, 2000, pp. 634–643.

[6] S. Bak, J. A. Cobb, and E. L. Leiss, "Load-balanced routing via bounded randomization," Department of Computer Science, University of Houston, Tech. Rep., Apr. 1999.

[7] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice Hall, 1992, ch. 5, pp. 464–475.

[8] L. N. Binh, N. M. Hoang, W. Zheng, C. Cieutat, and A. Sekercioglu, "Routing and wavelength assignment in GMPLS-based optical networks," in *SPIE Optical Transmission Systems and Equipment for WDM Networking II*, 2003.

[9] L. N. Binh, N. M. Hoang, W. Zheng, and A. Sekercioglu, "Modeling of wavelength routing and assignment in GMPλS-based DWDM optical networks," in *Conference on the Optical Internet & Australian Conference on Optical Fibre Technology*, 2003.

[10] T. Bonald and L. Massoulié, "Impact of fairness of internet performance," in *ACM Sigmetrics 2001*, 2001, pp. 82–91.

[11] C.-S. Chang and Z. Liu, "A bandwidth sharing theory for a large number of HTTP-like connections," in *Proceedings of INFOCOM 2002*, vol. 2, 2002, pp. 1310–1314.

[12] J. C. de Oliveira, C. Scoglio, I. F. Akyildiz, and G. Uhl, "New preemption policies for DiffServ-aware traffic engineering to minimize rerouting in MPLS networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 733–745, Aug. 1999.

[13] P. Dharwadkar, H. J. Siegel, and E. K. P. Chong, "A heuristic for dynamic bandwidth allocation with preemption and degradation for prioritized requests," in *21st International Conference on Distributed Computing Systems*, Apr. 2001, pp. 547–556.

[14] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: multipath adaptive traffic engineering," *Computer Networks*, vol. 40, no. 6, pp. 695–709, Dec. 2002.

[15] L. C. Hau, B.-H. Soong, and S. K. Bose, "Preemption algorithm with re-routing to minimize service disruption," in *Ninth IEEE Singapore International Conference on Communication Systems (ICCS'2004)*, Sept. 2004, pp. 521–525.

[16] P.-H. Ho and H. T. Mouftah, "Capacity-balanced alternate routing for MPLS traffic engineering," in *Seventh International Symposium on Computers and Communications (ISCC'02)*, July 2002, pp. 927–932.

[17] J. M. Jaffe, "Bottleneck flow control," *IEEE Transactions on Communications*, vol. 29, no. 7, pp. 954–962, July 1981.

[18] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, and A. Malis, "RFC 3212: Constraint-Based LSP Setup using LDP," Jan. 2002.

[19] K. Kar, S. Sarkar, and L. Tassiulas, "A simple rate control algorithm for max total user utility," in *Proceedings of INFOCOM 2001*, vol. 1, 2001, pp. 133–141.

[20] F. Kelly, "Chargin and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, Jan. 1997.

[21] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[22] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997, ch. 14, pp. 445–492.

[23] P. B. Key and D. R. McAuley, "Differential QoS and pricing in networks: where flow control meets game theory," *IEE Proceedings of Software*, vol. 146, pp. 39–43, Feb. 1999.

[24] S. Kunniyur and R. Srikant, "End-to-End congestion control schemes: Utility functions, random losses and ECN marks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 689–702, Oct. 2003.

[25] C. M. Largoa, H. Che, and B. A. Movsichoff, "Adaptive control algorithms for decentralized optimal traffic engineering in the internet," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 415–428, June 2004.

[26] K. Long and S. C. Z. Zhang, "Load balancing algorithms in MPLS traffic engineering," in *2001 IEEE Workshop on High Performance Switching and Routing*, May 2001, pp. 175–179.

[27] S. H. Low, "Multipath optimization flow control," in *Proceedings of IEEE International Conference on Networks (ICON 2000)*, 2000, pp. 39–43.

[28] S. H. Low and D. E. Lapsley, "Optimization Flow Control—I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[29] S. H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525–536, Aug. 2003.

[30] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proceedings of 1997 International Conference on Network Protocols*, Oct. 1997, pp. 191–202.

[31] Q. Ma, P. Steenkiste, and H. Zhang, "Routing high-bandwidth traffic in max-min fair share networks," in *ACM SIGCOMM96*, Aug. 1996, pp. 206–217.

[32] L. Massoulié and J. Roberts, "Bandwidth sharing: objectives and algorithms," in *IEEE Infocom'99*, 2001.

[33] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, Oct. 2000.

[34] E. Rosen, A. Viswanathan, and R. Callon, "RFC 3031: Multiprotocol Label Switching Architecture," Jan. 2001.

[35] E. Salvadori and R. Battiti, "A load balancing scheme for congestion control in MPLS networks," in *Eighth IEEE International Symposium on Computers and Communications (ISCC'02)*, July 2003, pp. 951–956.

[36] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1176–1188, July 1995.

[37] The Network Simulator — ns-2. The VINT Project. [Online]. Available: http://www.isi.edu/nsnam/ns/

[38] Z. Wang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sept. 1996.

[39] S. Zimmermann and U. Killat, "Resource marking and fair rate allocation," in *Proceedings of IEEE International Conference on Communications (ICC 2002)*, vol. 2, 2002, pp. 1310–1314.