TRAFFIC SCHEDULING AND LINK SHARING MECHANISMS FOR CONTROLLED LOAD SERVICE

By

CHIN YUN CHOONG

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA

In Partial Fulfillment of the Requirement for the Degree of Master of Computer Science

(Worth 6/30 credit hours)

June 2000

Abstract

The project studies and develops a simulator based on the link-sharing and traffic scheduling mechanism for Controlled Load service. The simulation adopts some of the features of the Class Based Queueing. The main factors that determine whether a packet scheduling algorithm is able to support integrated services or differentiated services on Internet is the ability to provide bandwidth guarantee. It is important to provide guarantee for minimum bandwidth when there are enough demands to ensure competing data flows do not degrade each others' performance. The objective of this project is to observe the performance of the link-sharing and traffic scheduling mechanism in providing bandwidth guarantee. Several experiments were carried out and can be divided into two main parts, the experiments with the "borrow" mechanism activated and the experiment without the borrow mechanism. These two parts were tested under different link usage and different bandwidth allocation to determine the efficiency of the "borrow" mechanism. We expect the number of dropped packets will decrease especially during congestion time when the "borrow" mechanism is activated. We also expect the mechanism can provide bandwidth guarantee when there are enough resources. The results indicate that, the link-sharing and traffic scheduling mechanisms are capable of providing the bandwidth guarantee and of reducing the numbers of dropped packets during congestion period by using the "borrow" mechanism.

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Dr. Mazliza Othman not only for her guidance in supervising my research but also for his sympathetic advice, ready encouragement and valuable suggestions throughout my project.

I also like to wish thank you to Mr. Richard Meyer from the UCLA for his guideline and advice. Special thanks to my friend Mr. Sim Kwan Yong for his help and support in finishing this project. A million thanks to my family for supporting me to study in this field.

Special thanks to my best friend Ng Seng Jge for her help, encouragement and support in my life and study.

Also thank to all my friends in University of Malaya especially those who are majoring in Computer Networking for supporting me throughout this course.

Chin Yun Choong
Faculty of Computer Science and Information Technology
University Malaya
Kuala Lumpur,
Malaysia

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

As traffic volume on the Internet increases rapidly day by day, and with the introduction of new real-time applications and services such as multimedia and multicasting applications, the traditional Internet protocols and services are inevitably not enough to support the requirements such as bandwidth needed by these applications. The standard Internet Protocol (IP) based networks were designed to provide a single level of service so called "best-effort", fair datagram delivery service.

The best-effort IP scales well as first, but as more hosts are connected, network service demands eventually exceed available capacity. Service is not denied, but its performance degrades gracefully. Although the resulting variability in delivery delays (jitter) and packet loss do not adversely affect typical Internet applications such as e-mail, file transfer and Web applications, applications with real time requirements such as those that deliver multimedia may face severe problems which is caused by delivery delays [1].

Since network congestion can potentially gives problem to real time data streams, mechanisms to guarantee Quality of Service (QoS) for those applications that require it need to be put into place. The objective of QoS is to provide some level of predictability and control beyond the current IP "best-effort service". Within these few years, a number of QoS protocols have been developed to satisfy the variety of applications needs. According to the Internet Engineering Task Force (IETF), there are two modules for providing QoS, they are Integrated Services (IS) and Differentiated Services (DS).

The Integrated Services (IS) model is based on a reservation-based traffic engineering assumptions. It reserves resources explicitly using a dynamic signaling protocol and employs admission control, packet classification and intelligent scheduling to achieve the desired QoS. An example of IS service is resource reservation.

[1] White Paper – QoS Protocols & Architectures, July 1999, www.qosforum.com

In [1], the authors use the term integrated services (IS) for an Internet service model that includes best-effort service, real time service and controlled link sharing. The latter services will be discussed in more details in another section.

Meanwhile, the DS model is based on reservation-less traffic engineering assumptions which classifies packets into a small number of service types and uses priority mechanisms to provide adequate QoS to the traffic. No explicit resource reservation or admission control is employed, although network nodes do have to use intelligent queueing mechanisms to distinguish traffic. An example of DS service is prioritization. The DS architecture is designed to provide a simple, easy to implement, low overhead tool to support a range of network services that are differentiated on the basis of performance.

Some of these service models are designed for use transparently within the network to meet different QoS requirements for different applications. Hence the idea of combining the traffic scheduling and link sharing mechanisms was developed and one of the approaches is the Class Based Queueing (CBQ) [2]. The CBQ is a strong candidate for a building block for introducing new Internet service models (from standardized Integrated Services [3] to newly proposed Differentiated Services [4,5]) because it provides :

- A degree of freedom for introducing a wide range of policies (based on services, protocol and network address information)
- Per "entity" traffic isolation, with a flexibility in defining "the entity", and therefore, the degree of aggregation (per flow, per user, etc)
- Bottleneck link sharing [6]

However there is a substantial level of complexity involved in the deployment of link sharing which subsequently leads to resource management mechanisms. Although the link-sharing mechanisms deployment is in progress but the development of resource management is still at an early stage and their effects on end-to-end performance are not

always straightforward, usually investigated only by simulations. Links to external networks especially in the Internet often must be shared between multiple organizations, protocols and traffic types. In these kind of situation, everyone involved in sharing the resources often wants some guarantee of bandwidth and this is where the link sharing mechanisms comes into the picture. Link sharing mechanisms introduces the notion of allowing multiple agencies, protocol families or traffic types to share the bandwidth of a given network link. It also allows each administrative domain control of their bandwidth.

By allowing isolation between the real time and best effort traffic in cooperation with the packet scheduling mechanisms that give priority to the real time traffic, controlled link sharing can also be a key component in enabling the deployment of priority-based packet scheduling algorithms designed to meet the end-to-end service requirements of real time traffic [2]. One benefit of link sharing is that it acknowledges the decentralized nature of the Internet and allows some local control of bandwidth distribution.

1.1 PROJECT OVERVIEW

In this project, we developed a simulation which contains the traffic scheduling and link sharing mechanisms for Controlled Load service. The simulation is mostly based on the CBQ approach and will be tested under several environments such as different utilization, different link bandwidth and different borrowing rate. The simulation will allocate link bandwidth to classes while assigning priorities to those classes. In this project, we have a high-priority real-time classes which use the controlled load service and a lower-priority non real time class which use the best-effort service.

Each class will have different bandwidth allocation and packets from the real-time class would receive priority scheduling as long as sufficient bandwidth is available, or they may use more than its allocated bandwidth if the process "borrow" is activated. This simulation made an assumption that all traffic are after the admission control procedure and has been accepted for controlled load service.

1.2 PROJECT OBJECTIVES

The services offered by established Internet protocols are still unsuitable for real time applications because variable queueing delays and packet losses due to congestion can occur at any time and any limit. Before real time applications such as remote video, multimedia conferencing, visualization and virtual reality can be used widely, the Internet infrastructure must be modified to support real time QoS to provide some control over end-to-end packet delays.

The real-time QoS is not only an issue for the next generation traffic management in the Internet but nowadays network operator are also requesting for the ability to control the sharing of bandwidth on a particular link among different traffic classes [2]. In an environment with limited bandwidth, fulfilling the demands of real time traffic requires a suite of real time services, including flow specifications for the real-time applications, a set-up procedure such as RSVP [7], and admission control procedures to control the number of admitted real time connections, in addition to a number of scheduling mechanisms at the gateway.

Therefore the objectives of this project are :

1) to develop a simulation based on some Class Based Queuing (CBQ) features which have link sharing and traffic scheduling mechanisms for Controlled Load services. This simulation have the traffic control capabilities that are required by the IETF's Integrated Services models. Therefore, it can be used as the traffic control for Controlled Load service with or without the admission control, or any resource reservation protocols.

2) to analyze the simulation under different situations such as limited bandwidth, different utilization and different "borrowing" scheme, which are permitted by the link sharing mechanisms.

3) to provide new resource management facilities for the gateway by using the link sharing mechanisms. The mechanism ensures that all the resources is fully utilized and thus, the wastage of resource can be decreased.

4) to provide gateways with the flexibility to accommodate emerging applications and network protocols by allowing different traffic types to share the bandwidth of a given network link.

1.3 SIGNIFICANCE OF THE RESEARCH

This project, hopefully will throw some light over the use of scheduling mechanisms like CBQ because it is expected that the intelligent scheduling mechanisms will play a key role in next generation multiservice IP networks [6]. Besides that, this project will give people the knowledge of link-sharing services. There has been an abundance of research about the needs of real-time traffic, but link-sharing services have received somewhat less attention in the research community.

The results of this project give an indication of the performance of link sharing mechanisms in providing resource management facilities for the gateway under different condition such as different utilization, limited bandwidth, and most important is the "borrowing" scheme. Besides that, the results of this simulation (which also using the same set of CBQ mechanisms) can be used to implement the traffic control capabilities that are required to implement the IETF's Integrated Services models.

Besides that, the result of this project can also be used as a reference to develop another simulation to test multiple agencies, protocol families or even organizations to share the bandwidth of a given network link. This simulation also works well with the reservation protocols such as RSVP by allowing "flows" to request reservations for higher-priority service.

1.4 SCOPE OF THE RESEARCH

This project will cover the development of a link sharing and traffic scheduling mechanisms simulation for Controlled Load service using the PaRallel Simulation Environment for Complex systems (PARSEC) [8]. PARSEC is a C-based discrete event simulation language and is discussed further in Chapter 3. The simulation consists of the source(s), destination(s), a packet scheduler and a link sharing component. In this project, we refer mostly to the CBQ mechanism because it outlines a set of flexible, efficiently-implemented gateway mechanisms that can meet a range of service and link-sharing requirements.

For the packet scheduler, we used the Weighted Round Robin (WRR) instead of Packet Round Robin (PRR) and for the link sharing scheduler, we will use the Top-Level link sharing. Different situations or conditions were tested in these simulation such as different link bandwidth or capacity, different link utilization (lightly loaded, loaded or heavily loaded) and finally, the different rate of "borrowing" scheme which plays an important role in the link sharing mechanisms. This simulation also studies the efficiency of the borrowing scheme under different situations and different link bandwidths.

The discussion regarding on CBQ, link-sharing mechanisms, controlled link-sharing and PARSEC is in Chapter 2 and Chapter 3.

1.5 THESIS ORGANIZATION

Chapter 2 covers mainly on the link-sharing mechanisms and approach to these mechanisms such as CBQ. This chapter also briefly compares the CBQ with the RSVP and explain in detail about the functions and features in the CBQ.

Chapter 3 highlights the analysis of the simulation especially the PARSEC and have a thorough look at the modeling of the simulation. Each of the simulation components and its

functions are discussed in detail. This is followed by chapter which highlights the implementation aspect of this project.

Chapter 5 discusses the testing of different situations and different bandwidth allocations faced by the simulation and the overall effects on this situation. Finally, chapter 6 concludes the overall presentation of the project and discusses future enhancements in this project.

Chapter 2

Literature Review

2.0 Introduction

The services offered by established Internet protocols are nowadays unsuitable for real-time applications due to variable queueing delays and packet loss which may lead to congestion at any time. Different solutions to the QoS over IP problem have been proposed and are currently under discussion in the IETF forum. The standardized Integrated Services (IS) and the newly proposed Differentiated Services (DS) are introduced to overcome the problems.

A service or QoS control service describes a named, coordinated set of QoS control capabilities that are provided by a single network element. A service definition must compromise a specification of the functions to be performed by the element, the information necessary to perform these functions, the information that the element exports to other entities in the system, as well as relevant information.

This chapter is organized as follows : Section 2.1 describes the Differentiated Service briefly while Section 2.2 gives an overview of the Integrated Service. Section 2.3 introduces the link-sharing mechanisms and its requirements. The following section will describe the link-sharing goals and some of the link-sharing structures. Section 2.4 gives an overview of Class Based Queueing (CBQ) which adopts and outlines a set of flexible and efficient gateways mechanism that can meet a range of service and link-sharing requirements. Section 2.5 describes briefly the Resource ReSerVation Protocol (RSVP) and RSVP goals. The following section discusses how RSVP works and the characteristics of RSVP. Section 2.6 discusses the relationship between CBQ and RSVP. The final section discusses the drawbacks of RSVP.

2.1 An Overview Of Differentiated Service (DS)

An alternative and simpler solution proposed by the IETF is based on Differentiated Services [9]. Differentiated Services adopt a purely datagram architecture which defines simple forwarding mechanisms for interior network routers, pushing most of the complexity onto the network edges. Traffic flows are divided in a few classes handled by routers according to a set of per-hop-behaviors (PHB). For example, at each router, frames of class A can be forwarded before frames of class B, or frames of class C can be dropped after frames of class D [10,11]. Each traffic class is identified by means of the Type Of Service (TOS) and DS bits defined in the IPv4 and IPv6 frame headers [12].

The main advantage of Differentiated Services is that it can offer Internet Service Providers (ISP) the capability to provide enhanced services in addition to the basic best effort service. The PHBs are the building blocks which can be combined to define end-to-end services according to a pre-defined Service Level Agreement (SLA). The IETF DiffServ working group distinguished between two different types of services, qualitative and quantitative [13], depending on the type of performance parameters offered.

Qualitative services only offer a relative guarantee, such as requiring the loss probability of class C to be low or lower than that of class D. Quantitative services provide concrete guarantees, for example requiring that a bound on the probability of a packet exceeding a delay threshold be met.

2.1.1 Scheduling Algorithms To Implement Differentiated Service

There are many ways or approaches to implement differential QoS within the network such as the Random Early Detection or commonly known as RED [14], but our main focus is in the routers queuing algorithm or scheduling discipline as the enabling mechanism. This is because we will use the mechanism called Class Based Queuing (CBQ) approach in this project which contains one of the scheduling algorithms together with the link-sharing mechanism.

2.1.1.1 FIFO Queuing

We will start with the FIFO queuing which is the base of best effort and single quality of network environments. FIFO queue has no inherent differentiation, or in more simple words, do not have any extra effort to differentiate the traffics by the router's transmission scheduler. Every packet which is scheduled to be transmitted on an output interface must wait all previously scheduled packets before transmission. All these packets have slots in a single per interface queue and when the queue is full, all subsequent packets arriving will be discarded until the queue become available once more. FIFO queue is a fair algorithm as with the basic Random Early Detection (RED) algorithm because it allocates the transmission resource fairly and imposes the same delay on all queued packets. To provide the differentiated services levels, it is necessary to change the fairness of this algorithm and introduce a mechanism to trigger preferential outcomes for classes of traffic.

Therefore, some changes or modifications have to be made to enable differentiated QoS and most basic modification of the single level FIFO algorithm is to divide traffic into number of categories. Then each category will be provided some resources according to a predetermined allocation structure by implementing some form of proportional resource allocation.

The Law of Conservation still holds here, such that the sum of the mean queuing delays per traffic category, weighted by their share of the resources they receive is limited, with the corollary that in reducing the mean queuing delay for one category of traffic will result in the increase in mean queuing delay for one or more of the remaining categories of traffic. For further information, refer to [15]. But, in practical one cannot improve the performance profile of one class of traffic without adversely affecting the performance profile of one or more of the other classes of traffic. The level of degradation will be similar in quantity to the level of improvement that was effected.

2.1.1.2 Priority Queues

Besides the modification above, another basic modification to the base FIFO structure is to create a number of distinct queues for each interface and associate a relative priority level to each. From there, packets are scheduled from a particular priority queue in FIFO order only when all queues of a higher priority are empty. This kind of model has an advantage of allowing the highest priority traffic to receive minimal delay, but on the other hand all other priority levels may experience resource starvation if the highest precedence traffic queue remains occupied. Thus, to ensure that all traffic receives some level of service, it is a requirement that the network uses admission policies which restricts the amount of traffic which is admitted at each elevated priority. Besides that, the scheduling algorithm is adjusted to ensure that every priority class receives some minimum level of resource allocation. The drawbacks of this simple priority mechanism are it does not scale well although it can be implemented with relatively little cost. In today's Internet, more sophisticated and more robust scheduling algorithms are required for QoS support.

2.1.1.3 Generalized Processor Sharing

This approach associates a relative weight or precedence to each individual traffic flow, and at every router segments each traffic flow into an individual FIFO queue and configure the scheduler to serve all queues in a bit wise round robin fashion. It allocates service to each flow in accordance with the relative weight [16]. This is an instance of a Generalized Processor Sharing or commonly known as GPS discipline. For more information, refer to [17].

2.1.1.4 Weighted Round-Robin and Deficit Weighted Round Robin

A basic approach is to use a packet's marked precedence to place the packet into a precedence-based queue and then use a weighted round robin scheduling algorithm to service each queue. The Weighted Round Robin employs weights proportional to a traffic

class bandwidth allocation. The weight determines the number of bytes that a traffic class is allowed to send in a scheduling round. If all packets are of identical size, this shows a relatively good approximation of GPS. But when the packets sizes different or vary, this algorithm may show significant deviation from a strict relative resource allocation strategy. Another algorithm called deficit weighted round robin [9] can be used to address this problem. The deficit weighted round robin modifies the round robin algorithm to use a service quantum unit. A packet is scheduled from the head of a weighted queue only if the packet size minus the per queue deficit counter is less than the weighted quantum value. The next packet in the queue is tested using a weighted quantum value which has been reduced by the size of the scheduled packet. When the test fails, the remaining weighted quantum size is added to the per queue deficit counter and the scheduler moves to the next queue.

Although this algorithm performs with an average allocation which corresponds to the relative weights of each queue, but it still shows unfairness within time frame which are commensurate to the maximum packet service time [16].

2.1.1.5 Weighted Fair Queuing

Weighted Fair Queuing or WFQ [18] tries to provide fairer resource allocation measures which protect conformance traffic sources from uncontrolled sources. WFQ attempts to compute the finish time of each queued packet if a bit-wise weighted GPS scheduler had been used. Then it schedules for service the packet with the smallest finish time which would have been receiving service in the corresponding GPS scheduler model. WFQ adopts a scheduling and packet drop policy where each packet drop is based on a preference for dropping packets with the greatest finish time in response to an incoming packet which requires a queue slot. Although WFQ needs a relatively complex implementation, it has its own advantages.

This scheduling algorithm does undertake fair allocation which does indeed ensure that

different categories of traffic are not capable of starving traffic in other categories. This algorithm also bounds the queue delay per service category which also provides a lever to create delay-bounded services without the need for resource reservation.

2.2 An Overview Of Integrated Service (IS)

The term integrated services (IS) for an Internet service model refer to the best-effort service, real-time service and controlled link-sharing service. Real-time traffic is characterized by a fixed or adaptive playback time (playback point) at the receiver; real-time packets arriving at the receiver after the playback time are discarded. Meanwhile, network operators are requesting the ability to control the sharing of bandwidth on a particular link among different traffic classes. They want to be able to divide traffic into a few administrative classes and assign to each a minimum percentage of the link bandwidth under conditions of overload, while allowing unused or excess bandwidth to be shared among each other. The important things here are that these classes may represent different user groups or different protocol families. This type of management facility is commonly called controlled link-sharing. The services that use this facility is called link-sharing service.

The requirements and mechanisms for integrated services have been the subjects of much discussion and research over the past several years. Many literature have been discussed such as in [19, 20, 21] etc and all this work has lead to the unified approach to integrated services support that is described in [22]. The core service model of [22] considers only quantitative service commitments concerning the bounds on minimum and maximum per packet transmission delays within a flow. Before we proceed, the term "flow" can be defined as a distinguishable stream of related datagrams that results from a single user activity and requires the same QoS. As an example, a flow might consists of one transport connection or one video stream between a given host pair. It is the finest granularity of packet stream distinguishable by the IS. A flow can be simplex, to have a single source but

R destinations. Therefore an R-way teleconference will generally require R flows, one originating at each side [22].

Applications can be divided into 2 categories according to the way their performance depends on the transmission delay behavior. These categories are real-time applications and elastic applications. As we mentioned earlier, the real time traffic is characterized by an adaptive or fixed playback time. Therefore, it is wise to look into details about playback applications. In a playback application, the source takes some signal, packetizes it and after that transmits the packets over the network. Along the way, the network unavoidably introduces some variation in the delay of the delivered packets. The receiver depacketizes the data and then attempts to play back the signal. This can be done by buffering the incoming data and then replaying the signal at some fixed offset delay from the original departure time. The term "playback time" refers to the point in time which is offset from the original departure time by this fixed delay. Data arriving after the playback point is useless in reconstructing the real-time signal. On the other hand, any data that arrives before its associated playback point can be used to reconstruct the signal.

This project focuses only on real-time applications. We start with a detailed discussion on real-time applications, and lastly the elastic applications in brief. In real-time applications which have been described above, the sensitivity to loss of fidelity leads to two application classes, each requiring a particular service class. The Intolerant applications which need the guaranteed service [23] to enforce a reliable upper bound on the maximum packet delay, and the Tolerant applications which need the predictive service.

2.2.1 Guaranteed Service

Guaranteed Service provides an assured level of bandwidth, a firm end-to-end delay bound and no queuing loss for conforming packets of a data flow. It is intended for applications with stringent real-time delivery requirements, such as certain audio and video applications that use "playback" buffers and are intolerant of any datagram arriving after their playback time [24].

Each router must characterizes the guaranteed service for a specific flows by allocating a specified bandwidth called R and buffer space, B, that the flow may consume by approximating the 'fluid model" of service [25,26.]. In a perfect fluid model, a flow conforming to a token bucket of rate r and depth b will have its delay bound by b/R provided $R \geq r$. C and D are the two error terms used to allow for deviations from the perfect fluid model in the router approximation. After introducing C and D the delay bound becomes b/R + C/R + D. However, with guaranteed service a limit is imposed on the peak rate, p, of the flow which results in a reduction of the delay bound. We also have to consider the maximum packet size, M for the packetization effect of the flow. The additional factors result in a more oresice bound to end-to-end queuing and the last formula for bounds on end-to-end delay is :

$$Q_{delayend2end} = (b - M)(p - R)/R(p-r) + (M + C_{tot})/R + D_{tot} \quad (case\ p > R \geq r) \qquad (1)$$

$$Q_{delayend2end} = (M + C_{tot})/R + D_{tot} \quad (case\ R \geq p \geq r) \qquad (2)$$

where C_{tot} and D_{tot} are summation of C and D error terms

For a router to initialize guaranteed service for a specific data flow, some information needed to be informed such as the T_{spec} which means traffic characteristics and R_{spec} which means reservation specification sent by receivers, to enable the router to calculate sufficient local resources to guarantee a lossless service requires the terms C_{sum} and D_{sum}, which represents the summation of the C and D error terms.

The parameters in T_{spec} are :

p = peak rate of flow (bytes/s)

b = bucket depth (bytes)

r = token bucket rate (bytes/s)

m = minimum policed unit (bytes)

M = maximum datagram size (bytes)

The parameters in R_{spec} are :

R = bandwidth i.e service rate (bytes /s)

S = slack terms (ms)

The main characteristic of Guaranteed Service is the traffic that uses this type of service must be policed at the network access points to ensure conformance to the T_{spec}. The usual enforcement policy is to forward nonconforming packets as best effort datagrams.

Guaranteed service also requires reshaping of traffic to the token bucket of the reserved T_{spec} at certain points on the distribution tree in addition to the policing of data flows at the edge of the network. Any packets failing the reshaping are treated as best effort and reshaping must be applied at any points where it is possible for a data flow to exceed the reserved T_{spec} even when all senders associated with the data flow conform to their

individual T_{spec}.

2.2.2 Controlled Load Service

On the other hand, the tolerant applications need the predictive service or another similar service called controlled-load service [27]. The controlled-load service tries to approximate the behavior of best-effort service under unloaded network conditions at any time. Client applications may assume that a very high percentage of the transmitted packets will be successfully delivered by the network, and that the transit delay experienced by a very high percentage of the delivered packets will not greatly exceed the minimum transit delay experienced by any successfully delivered packet [26]. Unlike guaranteed service, controlled-load service provides no firm quantitative guarantees.

A T_{spec} for the flow desiring controlled load service must be submitted to the router although it is not necessary to include the peak rate parameter. The router must make a commitment to offer the flow a service equivalent to that seen by a best effort flow on a lightly loaded network if the flow is accepted for controlled load service. The significance between guaranteed Service and Controlled Load service is that controlled load flow does not noticeably deteriorate as the network load increases. On the other hand, a best effort flow would experience progressively worse service such as higher delay or loss as the network load increased. Controlled load service usually works well for those classes of applications that can tolerate a certain amount of loss and delay provided it is kept to a reasonable level. A fine example of applications using controlled load service is adaptive real-time applications.

Routers that implement the controlled load service must check for conformance of controlled load data flows to their appropriate reserved T_{spec}. The important thing here is that any nonconforming controlled load data flows must not be allowed to affect the QoS offered to conforming controlled load data flows, or to unfairly affect the handling of best effort traffic.

The router should attempt to forward as many of the packets of the non conforming controlled load data flow as possible and this can be done by dividing the packets into 2 groups, conforming and nonconforming groups. The nonconforming groups are forwarded on a best effort basis. Another way is the router may choose to degrade the QoS of all packets of a nonconforming controlled-load data flow equally.

After discussing the real time applications in detail, now we briefly discuss the elastic applications. Elastic applications will always wait for data to arrive and it is not that these applications are insensitive to delay which can often harm the application's performance by increasing the delay of a packet. Rather, the elastic applications typically uses the arriving data immediately rather than buffering it for some later time. It will always choose to wait for the incoming data rather than proceed without it. This type of applications do not require any a priori characterization of the service in order for the application to function because arriving data can be used immediately. The performance of elastic applications will depend more on the average delay than on the tail of the delay distribution if it is given distribution of packet delays. Subcategories of elastic applications with different delay expectations are interactive burst such as (Telnet, X, NFS), interactive bulk transfer (FTP) and asynchronous bulk transfer (electronic mail, FAX).

An appropriate service model for elastic applications is to provide "as-soon-as-possible" or ASAP service. Best effort service is not subject to admission control.

2.3 Introduce to link-sharing mechanism

As discussed early, the link-sharing mechanism presents the gateway with a new functionality for resource management in the packet networks. There are three requirements for link-sharing and the first one is to share bandwidth on a link between multiple organizations, where each organization wants to receive a guaranteed share of the link bandwidth. Besides that, when the bandwidth is not being used by one organization, it

should be available to other organizations sharing the link. The second requirement is to share bandwidth on a link between different protocol families such as IP and SNA, where controlled link-sharing is desired because the different protocol families have different responses to congestion. The last requirement for link-sharing is to share bandwidth on a link between different traffic types such as telnet, ftp or real time audio and video.

2.3.1 Link-Sharing Goals

This section briefly discusses about the link-sharing goals. The requirements for link-sharing are the same no matter if the link-sharing is between service classes, protocols families, organization or traffic types. First of all, the link-sharing structure specifies the desired policy in terms of the divisions of bandwidth in a particular link especially during congestion. There are many kinds of link-sharing structure for simulation such as flat link-sharing, two-agency link-sharing, three-agency link-sharing and hierarchical link-sharing.

See figures below for more details.

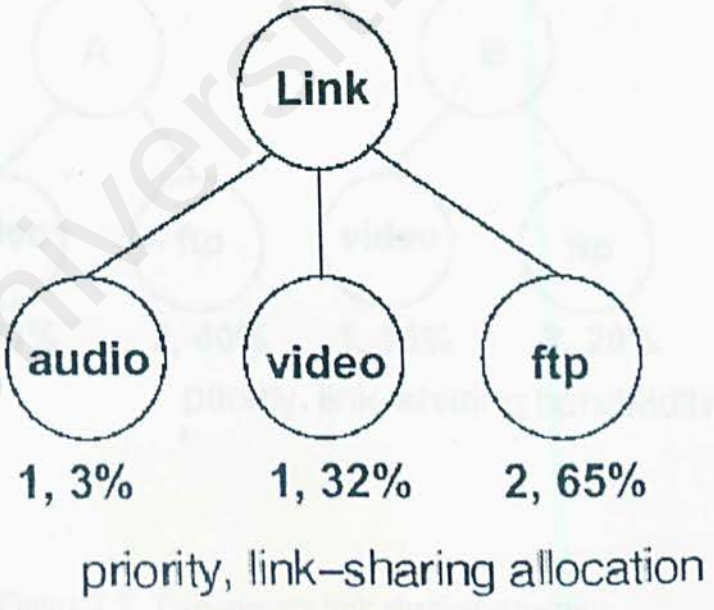


Figure 2.1 : Flat link-sharing structure

This figure is extracted from [2]

Note : Figure 2.1 shows that the link has three classes which are audio, video and ftp. Audio and video class have priority 1 while the ftp class has priority 2. Therefore, the audio and video class have a higher priority than the ftp class. This can be seen by the number on the left side of each class which represents the priority. The percentage on the right hand side of each class represents the link-sharing allocation. Class audio has been given 30% of the overall bandwidth while the video class has 32% of the overall bandwidth. The link allocates 65% of the overall bandwidth to class ftp.

Figure 2.1 to figure 2. 4 show examples of priority allocated to each class based on class importance.

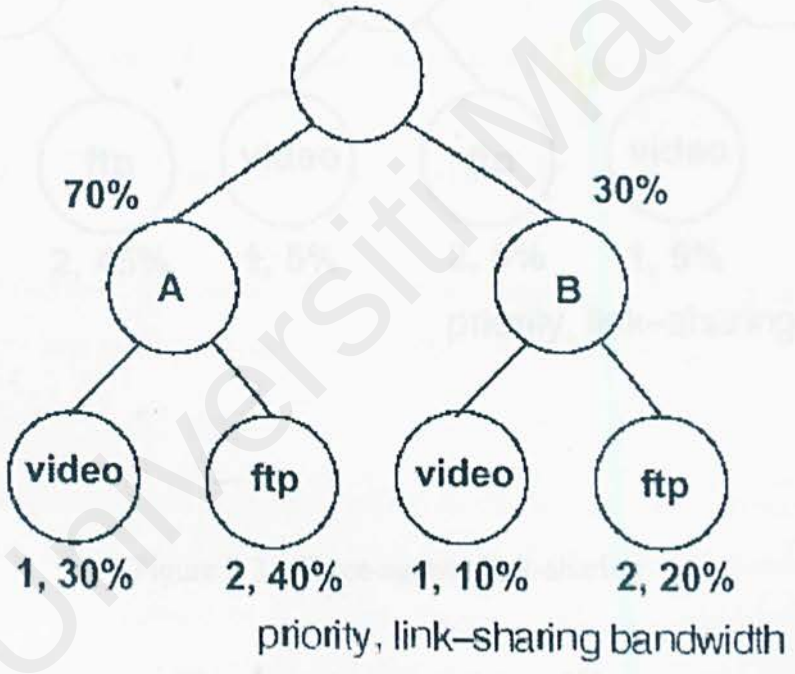


Figure 2 2: Two-agency link sharing structure

The figure is extracted from [2]

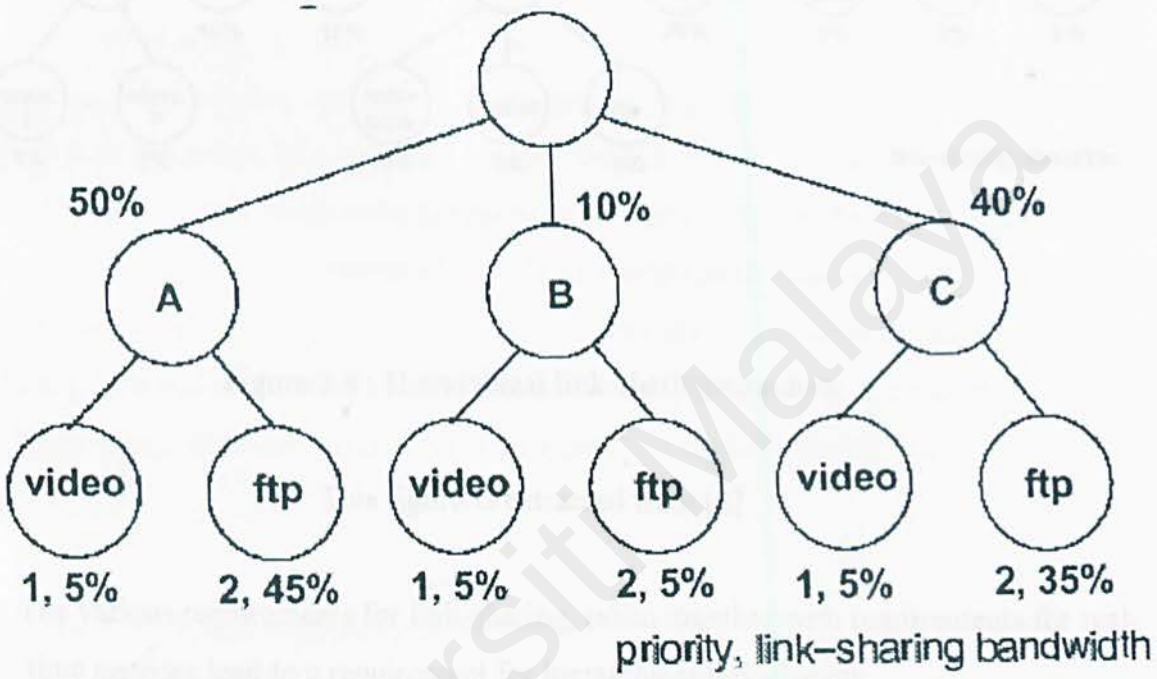


Figure 2 3 : Three-agency link-sharing

This figure is extracted from [2]

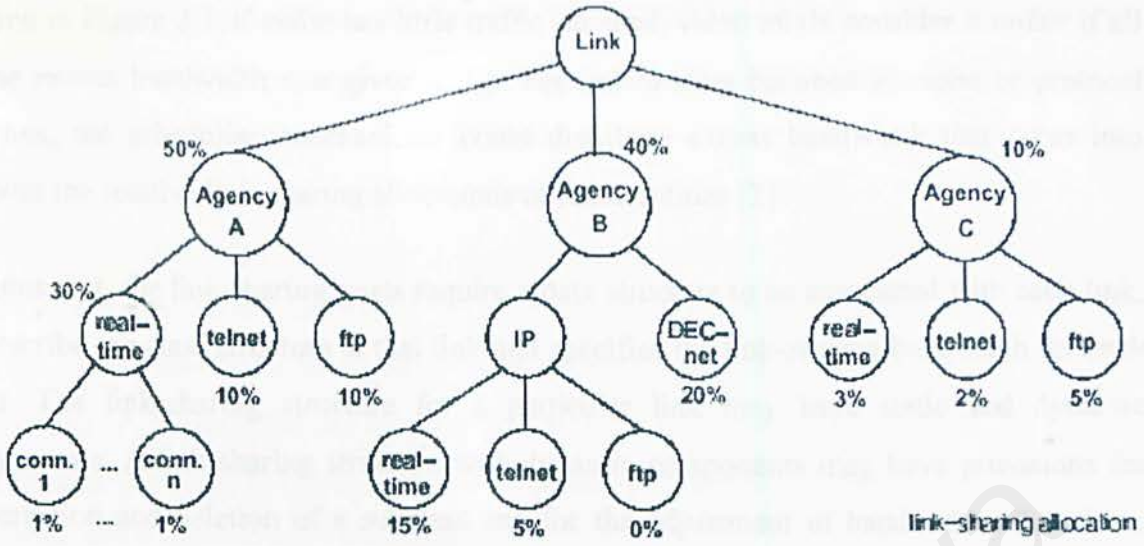


Figure 2.4 : Hierarchical link-sharing structure

This figure is extracted from [2]

Note : The various requirements for link-sharing, taken together with requirements for real-time services lead to a requirement for hierarchical link-sharing

The first link-sharing goal is that each class with enough demand should receive roughly its allocated bandwidth over some interval of time, especially during the congestion periods. Some classes might be restricted to their link-sharing bandwidth during the congestion times due to this link-sharing goal. Take note that, the bandwidth received by a class with a link-sharing allocation of zero is determined by other scheduling mechanisms; the link-sharing mechanisms do not guarantee any bandwidth to this class in times of congestion.

The secondary link-sharing goal is that if a class does not fully utilize its allocated bandwidth, the distribution of the excess bandwidth among the other classes should not be

arbitrary but should follow an appropriate set of guidelines. As an example, consider link-sharing in Figure 2.1, if audio has little traffic to send, video might consider it unfair if all of the excess bandwidth was given to ftp. For link-sharing between agencies or protocol families, the scheduling mechanisms could distribute excess bandwidth that takes into account the relative link-sharing allocations of those entities [2].

Besides that, the link sharing goals require a data structure to be associated with each link, to describe the class structure at that link and specifies the link-sharing bandwidth for each class. The link-sharing structure for a particular link may have static and dynamic components. A link-sharing structure with dynamic components may have provisions for the creation and deletion of a subclass and for the adjustment of bandwidth allocations. This kind of link-sharing mechanism is best used to monitor the bandwidth of specific real-time traffic flows so that the real-time flows do not monopolize the bandwidth on the link. As for the static link-sharing structure with fixed classes and bandwidth allocations, the link-sharing bandwidth allocated to each agency is set by the network administrator. Such a static link-sharing structure would be appropriate for a link shared between multiple agencies.

2.4 An Overview of Class Based Queueing

Class Based Queueing (CBQ) is not only a scheduling mechanism that provides link sharing between agencies or organization that are using the same physical link but also at the high level CBQ is a link-sharing resource manager. As a link-sharing mechanism, CBQ provides the gateway with a new functionality for resource management. CBQ is a proposal by Sally Floyd and Van Jacobson of the Lawrence Berkeley Laboratories to provide bandwidth control with the link sharing mechanism at the gateway level [2].

The most important feature in CBQ is that it presents the notion of allowing multiple agencies, protocol families, or traffic types to share the bandwidth of a given link. This is an improvement because link sharing guarantees that any excess bandwidth resulting from

an agency that is not fully utilizing its share is redistributed to other agencies, thus improving link utilization [28]. In addition to that, using the same set of link-sharing mechanisms, CBQ implements the traffic control capabilities that are required to implement IETF's Integrated Services models.

Hierarchical link-sharing, one of the most powerful features of CBQ allows each agency to assign its own bandwidth to different types of traffic, allocating the right share to each one. The unused bandwidth of an agency's class is distributed first to its leaf classes instead of other agencies. To achieve the situation above, the CBQ framework saturates the problem of resource management into 2 types of scheduling functions, that are a link sharing scheduler and a generalized packet scheduler.

The CBQ operation is based on the interaction between the general scheduler and the link sharing scheduler. The link-sharing scheduler is responsible for maintaining link-sharing constraints within a link-sharing structures while the generalized packet scheduler will guarantee the appropriate service to each class, distributing the bandwidth according to their allocations.

2.4.1 The Function Of Link-Sharing Scheduler

The link sharing scheduler in CBQ uses the top-level link-sharing scheduler approach to approximate the formal link-sharing scheduler without the computational complexity of formal link-sharing. Top-level link-sharing uses a heuristic called "top-level" to maintain link-sharing constraints. This "top-level" heuristic refers to the depth of the link-sharing structure to which a given traffic class may borrow bandwidth. Rules for setting the "top level" heuristic can be found in [2]. The use of Top-Level instead of Ancestor-Only link sharing [2] allows a class to receive its allocated bandwidth more accurately.

2.4.2 The Function Of Packet Scheduler

The generalized packet scheduler can be anything ranging from a simple Packet Round Robin (PRR) to a more sophisticated Weighted Round Robin (WRR). In both cases, the scheduling algorithms employ priority based scheduling. Packets are sent from the highest priority level first. The difference between WRR and the PRR is regarding how the packets are scheduled within a priority level. For PRR, plain round-robin scheduling arbitrates between traffic classes in the same priority level. The WRR scheduler differs in that it employs weights proportional to a traffic class bandwidth allocation. The weight determines the number of bytes that a traffic class is allowed to send in a scheduling round.

In [2], the gateway maintains a separate queue for each class associated with the output link. After each packet is transmitted on the output link, the general scheduler decides which class can send a packet on the link next. The general scheduler schedules packets from higher-priority classes first. Within classes of the same priority, the general scheduler uses a variant of weighted round-robin, with weights proportional to the bandwidth allocations of the classes. The weights determine the number of bytes that a class is allowed to send at each round. When a class sends more than its allocated number of bytes (because packets are not broken into byte-sized pieces), that class's byte-allocation for the following round is correspondingly reduced.

The use of weighted round-robin to service classes of the same priority level serves two functions. The first function is to ensure that each priority-one class receives its allocated bandwidth even over fairly small time intervals. If at most half of the link bandwidth is allocated to priority-one classes, then each priority-one class with sufficient demand is guaranteed to receive at least its allocated bandwidth in each round of the round-robin.

The second function of the weighted round-robin is to ensure that bandwidth is distributed to unregulated classes of the same priority in proportion to the bandwidth allocations of those classes. The distribution of the `excess' bandwidth among the other classes should not

be arbitrary, but should follow some appropriate set of guidelines. The use of a priority-based general scheduler with weighted round-robin within priority levels results in "excess" bandwidth being distributed by the general scheduler to the higher priority classes, with the distribution proportional to the relative link-sharing allocations of those classes.

2.5 The Resource reSerVation Protocol (RSVP)

The Resource reSerVation Protocol, or RSVP for short, is a mechanism for reserving bandwidth for end-to-end connections. It is designed to work with integrated services, but can also be used with other service models. By reserving bandwidth along the path, we are able to guarantee the amount of allocated bandwidth along a specific path in the network, and therefore guarantee the quality of service for the associated data stream traversing this path. The term "resource" mostly refers to something of value in a network infrastructure to which rules or policy criteria are first applied before access is granted. Two good examples of resource are bandwidth on a network and buffers in a router.

RSVP is designed with the following goals in mind [7] :

- Accommodate heterogeneous receivers.
- Adapt to changing multicast group memberships.
- Allow receivers to switch channels
- Adapt dynamically to changes in the underlying unicast and multicast routes.
- Exploit the characteristics of the resource requirements of different applications in order to utilize network resources efficiently.

- Make the design modular to accommodate heterogeneous underlying network technologies.

- Control the protocol overhead so that the complexity grow less than linearly with the number of participants.

2.5.1 How Does RSVP Work ?

We start with a request for bandwidth and each request is made of two parts : the *flowspec* and the *filterspec*. The *flowspec* and *filterspec* form a flow descriptor. The filterspec together with a session specification, defines the set of data packets - the flow – to receive the QoS defined by the *flowspec*. So the *flowspec* simply specifies the QoS that is desired and is used to set parameters in the node's packet scheduler, or other link layer mechanism. Meanwhile, the *filterspec* is used to set parameters in the packet classifier.

The admission control is then used to check the request to determine whether there is even enough bandwidth available and the policy control is used to check if the requestor has the necessary privileges to allocate bandwidth. Once all the checks are passed, an allocation is made and the request is sent to the next hop in the chain of routers connecting the sending and receiving hosts. After the request successfully makes it to the destination, a message is sent back to the originating process to indicate the allocation was successful. Otherwise, all allocations up until the point of failure are undone and a failure message is sent back to the requestor.

Along each hop, the path is recorded so that all packets sent through the reservation are guaranteed to go through routers that know about and respect the guarantee. A tear-down message allows all intermediate routers to release their respective bandwidth allocation when the hosts are done with the reserved connection. See figure 2.5 for more details.

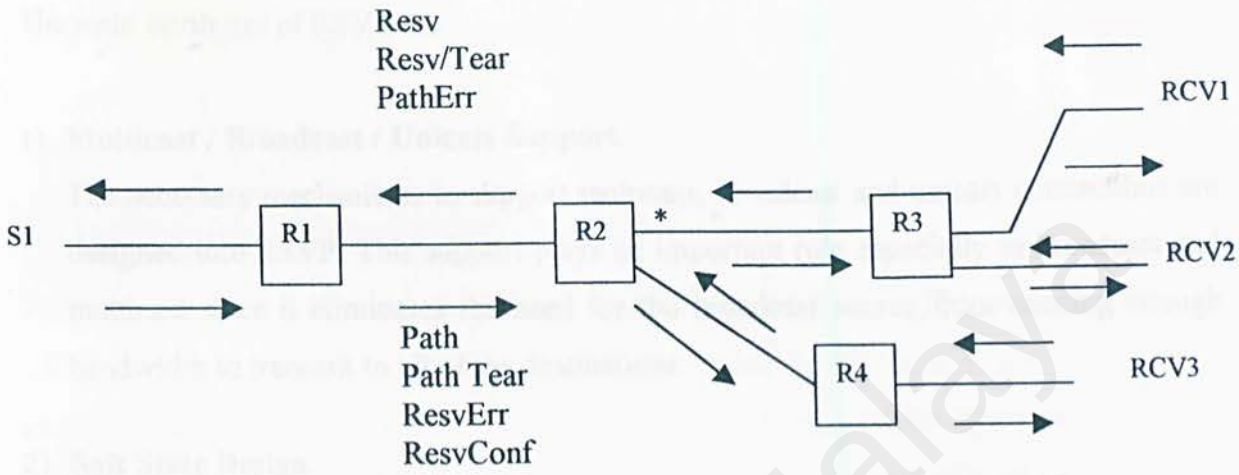


Figure 2.5 Direction of RSVP flow

The figure is extracted from [24]

Figure 2.5 shows an example of RSVP for a multicast session involving one sender, S1 and three receivers, RCV1-RCV3. The main messages used by RSVP are the Path message which originates from the traffic sender and the Resv message which originates from the traffic receivers. The main functions of the Path message are to install reverse routing state in each router along the path and to provide receivers with information about the

characteristics of the sender traffic. On the other hand, the function of the Resv messages is to carry reservation requests to the routers along the distribution tree between receivers and senders.

2.5.2 Key Attributes of RSVP

The main attributes of RSVP are :

1) **Multicast / Broadcast / Unicast Support**.

   The necessary mechanisms to support multicast, broadcast and unicast connections are designed into RSVP. This support plays an important role especially in broadcast and multicast since it eliminates the need for the broadcast source from needing enough bandwidth to transmit to all of the destinations.

2) **Soft State Design**.

   In RSVP, a soft state is used instead of hard state design. The soft state means that all connections are established and torn down dynamically. Bandwidth allocation and provision is based on a per request basis rather than hard allocating amounts for certain types of streams.

3) **Support for Routers That Don't Support RSVP**.

   RSVP can scale to very large multicast group but the designer of RSVP realized that RSVP would not be available at all locations immediately. With this in mind, it is, therefore, possible that reservations could be made through a network that doesn't support RSVP. In such cases, a guarantee is made by those that do support RSVP, but only best effort is supported during transmission through non-RSVP clouds. The important point here is that connections are not denied in the event a non-RSVP cloud is encountered.

4) **Graceful Failure**.

Hosts with a reservation must generate periodic RSVP "keep-alive control messages". These messages are read by the routers along the path to the destination host informing them that the connection is still in use. By generating these keep-alive messages, routers can automatically shutdowns a connection when it doesn't see a message within a pre-determined period. This keeps hosts which have failed from holding onto a reservation.

2.6 The Relationship between CBQ and RSVP

It is common to perceive that RSVP is a competitor of CBQ. [29] stated that CBQ and RSVP are orthogonal. CBQ does not need or require RSVP or any other reservation protocol and is concerned with how a router handles arriving packets. Their relationship is just like requester and performer. RSVP only requests for bandwidth while CBQ only performs the tasks to guarantee the bandwidth assignment once it has been allocated.

An example where CBQ can be used without any reservation protocol for a router is where separate allocations of link bandwidth is made to different classes of traffic as defined at the router. In the absence of reservation protocols such as RSVP, network administrators could made full use of information in the IP and TCP packets headers (e.g IP source and destination addresses, protocol fields, TOS bits) or any information about the arriving interfaces at the router to decide which packets to classify to the higher-priority classes.

If CBQ is used with reservation protocols, "flows" could dynamically request reservations for higher-priority service. In such situation, CBQ cooperate well with RSVP. One fine example would be to have a single high-priority class for Controlled Load traffic, for flows to use RSVP as the reservation protocol. The router will use the admissions control algorithms, statistical multiplexing of Controlled Load traffic in a single class and FIF0 queueing within the Controlled Load class [29].

At the other end of the spectrum but with more scaling problems at very large scales, CBQ could be used with RSVP for Guaranteed Service. A new CBQ class will be created at the router for each flow that is granted guaranteed service [29].

2.7 RSVP greatest shortcomings

In terms of potential deployment in the global Internet, RSVP's greatest or main drawbacks have to do with the question of scalability and of pricing structure. The questions of scalability have to do with the state that would be required in the network for RSVP on links with very high levels of statistical multiplexing. On the other hand, the question of pricing structures, which is perhaps more fundamental, comes from the fact that a flow is not likely to receive "special" treatment from all of the routers along a path in the global Internet unless those routers have some incentives, such as differential pricing, to grant that special treatment.

Similarly, "special" treatment for some traffic is not likely to work unless users have some incentives. But again, topic such as differential pricing will be asked and, thus, makes it difficult to ask for special treatment for all of their traffic. It is hard to envision the viable, practical, low-overhead pricing structure that would enable the deployment of RSVP in the global Internet [29].

Besides that, security issues arise when RSVP scales to very large multicast groups because it uses receiver-oriented reservation requests that merge as they progress up the multicast tree. This must be resolved to ensure that unauthorized sources do not make fake resources.

In conclusion of this section, the current form of the RSVP specification is only appropriate for multimedia applications run in small, private networks, are the most likely applications to benefit from the deployment of RSVP. Due to its inadequacies of scaling and lack of policy control, RSVP may be more manageable within the extent of a smaller, more controlled network environment than in the global Internet.

Chapter 3

Simulation Design And Modeling

3.0 Introduction

This simulation is written using the PARSEC [8] simulation language. As mentioned earlier PARSEC is a C-based discrete event simulation language which uses the process interaction approach to discrete-event simulation. Interaction among the events are modeled by timestamped message exchanges among the corresponding logical processes. In PARSEC, an object or a set of objects is represented by a logical process.

The simulator consists of three main parts which are :-

1) a driver entity,
2) a CBQ approach server entity and
3) source (s) / destination entities.

The driver entity is responsible for creating the rest of the entities and for distributing the communication topology to other entities after the creation of these entities. The CBQ approach server entity consists of link-sharing scheduler, a packet classifier, a borrower, a dustbin and a packet scheduler entity. Each of these components function is discussed in detail in the following section. We will only take some important features of the CBQ for this simulation and that is why we call the server the CBQ approach.

The rest of this chapter is structured as follows: in section 3.1 we talk about some key terms which is necessary to understand CBQ and also discuss thoroughly about the function of each components in the CBQ approach server entity. Section 3.2 describes the simulation language in details which the simulation methodologies is based on. This is followed by the description of the modeling or architecture of the simulator and also the operation flow of the simulation in the final section.

3.1 Simulation Components Function

In this simulation, the major components are a packet classifier, a link-sharing scheduler, a packet scheduler, a borrower and a dustbin. Before we proceed, we should examine the term classes and "borrowing" to understand the components function in more detail.

Classes

Each packet which enters the system is classified based on its route and/or its type of service. Each class maintains its own queue of packets and is unaware of demands on the system by other classes in the system. It is up to the scheduler to determine whether the next packet to be transmitted should come from a particular class.

Borrowing

When a class runs out of resources, it may utilize the bandwidth from sibling classes that are not using all of the bandwidth that have been allocated to them. If all of the bandwidth from sibling classes have been utilized, a class may start using bandwidth from its parent's siblings so long as they too are not using it. This process is called borrowing. We use the flat link-sharing structure in this simulation and there are only two classes or child classes and without any parent's siblings since these two classes come from one root. We use the flat link-sharing structure because there are only two types of traffic in this simulation which are the controlled load service traffic and the best effort service traffic, as proposed in [29] and to see the efficiency of the borrowing mechanism in cooperating with reservation protocols.

3.1.1 Packet Classifier

In order to support traffic control and accounting, each incoming packet is first identified through the information gained by the filter specs as belonging to a particular flow, and

then mapped into a corresponding service class. It is important to note that all packets within one class are treated equally.

The selection of the appropriate class is determined by each router and reflects the importance that a router gives to an individual flow. Among ways to classify the packets are by looking at the source and destination host address, the protocol number and the port fields.

In order to reduce overhead, this simulation makes use of an additional flow-id field in the packet header to allow a short-cut classification of the packets.

3.1.2 Link-Sharing Scheduler

Scheduling algorithms decide which enqueued packets are chosen to be delivered and in what order. CBQ does not explicitly define which algorithm should be used, but they are able to support two modes of operation: general scheduling and link-sharing scheduling. Link-sharing scheduler plays a main role when the traffic classes are diverse and when the requested bandwidth capacity is higher than available bandwidth.

In CBQ, the top-level link-sharing scheduler is used to approximate that of the formal link-sharing scheduler without the computational complexity of formal link-sharing. Due to the complexity of the link-sharing scheduler, we will not fully use all the features in the link-sharing scheduler in this simulation, the link-sharing scheduler consists of two first in first out servers and a new component called a borrower. The major features of a link-sharing scheduler such as sharing of excess bandwidth with other classes, and allowing the allocation of link bandwidth between classes are retained in this simulation. We will be using the link-sharing structure of flat link-sharing (Figure 2.1) and it consists of two classes which are Controlled Load and Best Effort.

3.1.3 Packet Scheduler

The packet scheduler main function is to manage the forwarding of different packet streams by reordering the output queue according to the importance of the packets' classes. Therefore, the packet scheduler must be a part of the output driver of an operating system, namely the part which manages the output packet queues. There are many packet scheduling algorithms eg: priority-based, round robin or weighted-fair-queueing (WFQ). The packet scheduling may also include deciding which packets to drop in case of an overload. It also may act as a traffic estimator by measuring the properties of the outgoing packet streams. This information is used to generate traffic statistics which supports packet scheduling and admission control decisions.

In CBQ, two generalized packet scheduler, Packet-by-packet Round Robin (PRR) and Weighted Round Robin (WRR) are used. In both cases, the scheduling algorithms employ priority based scheduling. The priority based scheduling sends packets from the highest priority level first. The PRR differs from the WRR with regard to how the packets are scheduled within a priority level. The WRR scheduler employs weights proportional to a traffic class bandwidth allocation. The weight determines the number of bytes that a traffic class is allowed to send in a scheduling round. For PRR, plain round-robin scheduling arbitrates between traffic classes with the same priority level.

Since in this simulation, there are only 2 classes i.e. Controlled Load and Best Effort, and the link-sharing structure for this simulation is a flat link-sharing which has only one root class and 2 child classes without any leaf classes, the use of any two scheduling algorithms above is the same. This is because the Controlled Load traffic is usually of a higher priority than the Best Effort traffic. Thus, the packets from the Controlled Load traffic are served first. We choose the WRR because [2] stated that WRR has two advantages over PRR scheduling within a priority level. WRR gives better worst case delay behavior than PRR scheduling for higher priority classes and WRR scheduling algorithm allows excess bandwidth to be distributed among classes in a priority level according to the bandwidth allocations of those classes.

The packet scheduler here will send packets according to their priority with the highest priority first which is the Controlled Load traffic.

3.1.4 Borrower And Dustbin

The borrower allows the Controlled Load traffic to borrow excess bandwidth from the Best Effort traffic. The borrower will fix a borrowing rate and the Controlled Load traffic cannot borrow more than the rate to avoid it from monopolizing all the bandwidth. When Best Effort wants to use the allocated borrow bandwidth, the Controlled Load service will have to return the borrowed bandwidth to the Best Effort traffic. The important thing here is that the Best Effort class cannot borrow from the Controlled Load class since that class has higher priority and more important information. The function of the dustbin is to store all packets dropped by the entities and keep a record of the dropped packets.

3.2 PARallel Simulation Environment for Complex systems or PARSEC

It is necessary to know the facilities and the operation provided by PARSEC before we proceed to the simulation design and modeling. The PARSEC language is based on C and a PARSEC programs consist of entities which exchange messages. The following section describes these two features and other important features.

3.2.1 Entities

A PARSEC program is a collection of C functions and entity definitions. An entity definition describes a class of objects. In PARSEC, an object is also referred to as a physical process, or a set of objects in the physical systems is represented by a logical process. Interactions among physical processes or sometimes called events are modeled by timestamped message exchanges among the corresponding logical processes [8]. Instances of an entity type may be created to model objects in the physical system.

The definition of an entity type is exactly same to the definition of a C function. It consists of a heading which is similar to an ANSI function heading and it specifies a name for the entity type and gives a list of typed parameters. Each entity also consists of a body which is a compound statement that describes the actions executed by an entity type. The body of an entity is terminated by a **finalize** block.

A PARSEC entity is created by using **new** statement and the execution of a new statement returns a unique identifier of type **ename. Ename** is used to store the entity identifiers and an entity may refer to its own identifier by using the keyword **self**. A PARSEC entity may terminate itself by executing a C return statement or by falling off the end of the entity body.

3.2.2 Message Communication

Entities communicate with each other through buffered message passing. Each entity is provided with a unique message buffer. Asynchronous send and receive primitives are provided to respectively deposit and remove the messages from the message buffer of an entity. PARSEC uses typed messages and a message-type consists of a name and a parameter list[8].

In PARSEC, a message is defined syntactically similar to the declaration of a C struct and message type declarations are treated as global. A message-type may consist an empty parameter-list and it is usually used to define signals or acknowledgement.

An entity sends a message to another entity using a **send** statement. The **send** statement performs an asynchronous send. It means the sending entity copies the message parameters into a memory block and delivers the message to the underlying communication network and resumes execution. The unique features of the message is that every message is implicitly timestamped with the current value of the simulation clock.

An entity accepts messages from its message buffer through executing a **receive** statement. If the message buffer contains exactly one enabling message, the message then is removed from the buffer and delivered to the entity in a specified variable. When a message is received, the internal clock of the entity will advance to the greater of the time specified on the timestamped of the message or the current time of the entity. Therefore, the entity's clock moves forward. For further details about the syntax of the message, please refer to [8].

3.2.3 Time Management In PARSEC

As discussed earlier, each message involved in the simulation uses a timestamp and interactions among physical processes or events are modeled by timestamped message exchange among the corresponding logical processes.

Thus, PARSEC introduces a timeout clause and if a receive statement executed by an entity includes a timeout clause with wait time *tc*, execution of the statements schedules a timeout for the entity. In PARSEC, there are two different ways of handling simultaneous events through the use of the **in** and **after** keywords at the timeout clause. For further details, please refer to [8].

Besides that, in order to suspend an entity unconditionally for a specified duration to enable activities like serving a request, a **hold** statement has been introduced. When the statement **hold**(*delay-time*) is executed, the simulation clock of the entity will advance by *delay-time* time units.

In conclusion, one of the important distinguishing features of PARSEC is its capability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures. According to [8], PARSEC is designed to cleanly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute it.

In the next section, we shall look at the design and modeling of the simulation. As we discussed earlier, the simulation used some of the CBQ features in this project.

3.3 Simulation Design And Modeling

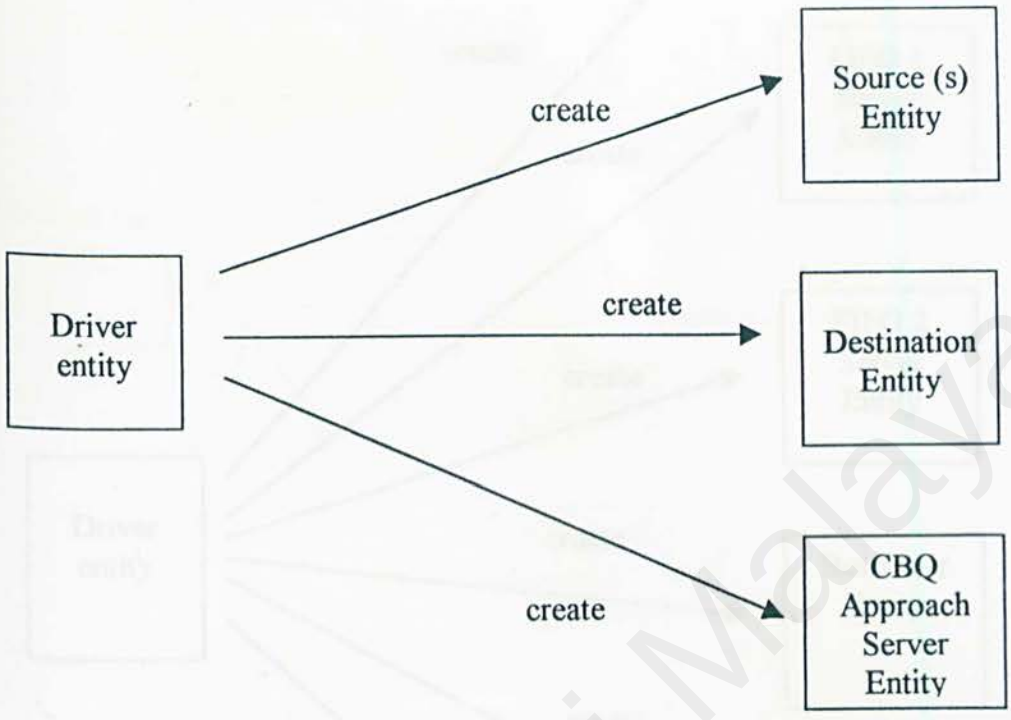


Figure 3. 1 : Creation Of New Entity Using Driver Entity

The first step of the simulation is the creation of new entities by the driver entity. The driver entity serves a purpose similar to the main function of a C program. Execution of a PARSEC program is initiated by executing the first statement in the body of entity driver. The CBQ approach server contains six entities, i.e a packet classifier, two FIFO servers, a borrower, a dustbin and a packet scheduler. The driver entity must create all the six entities at the beginning of the simulation. Please refer to Figure 3.2 for more details.

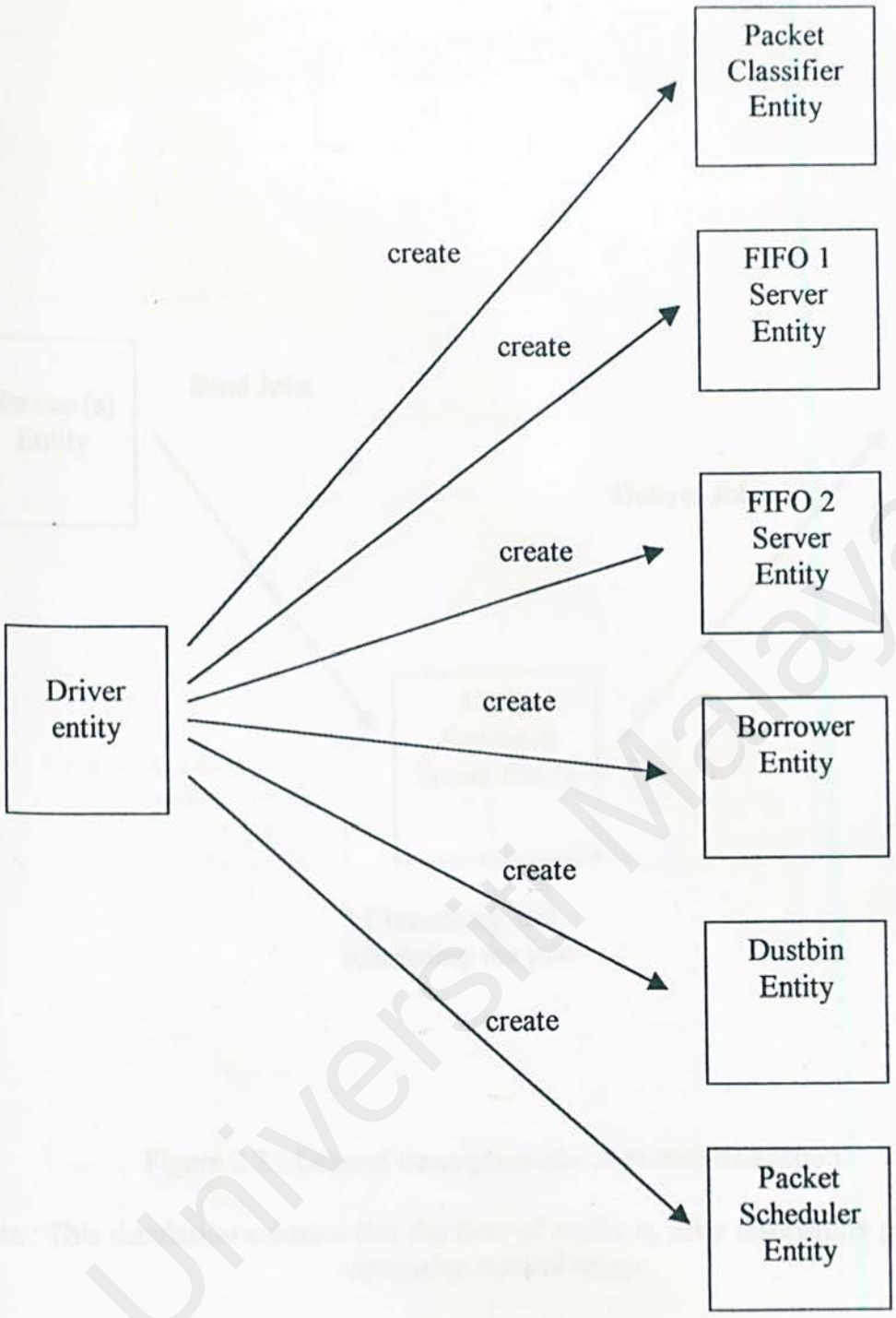


Figure 3.2 : Creation Of New Entities In CBQ Approach Server Using A Driver Entity

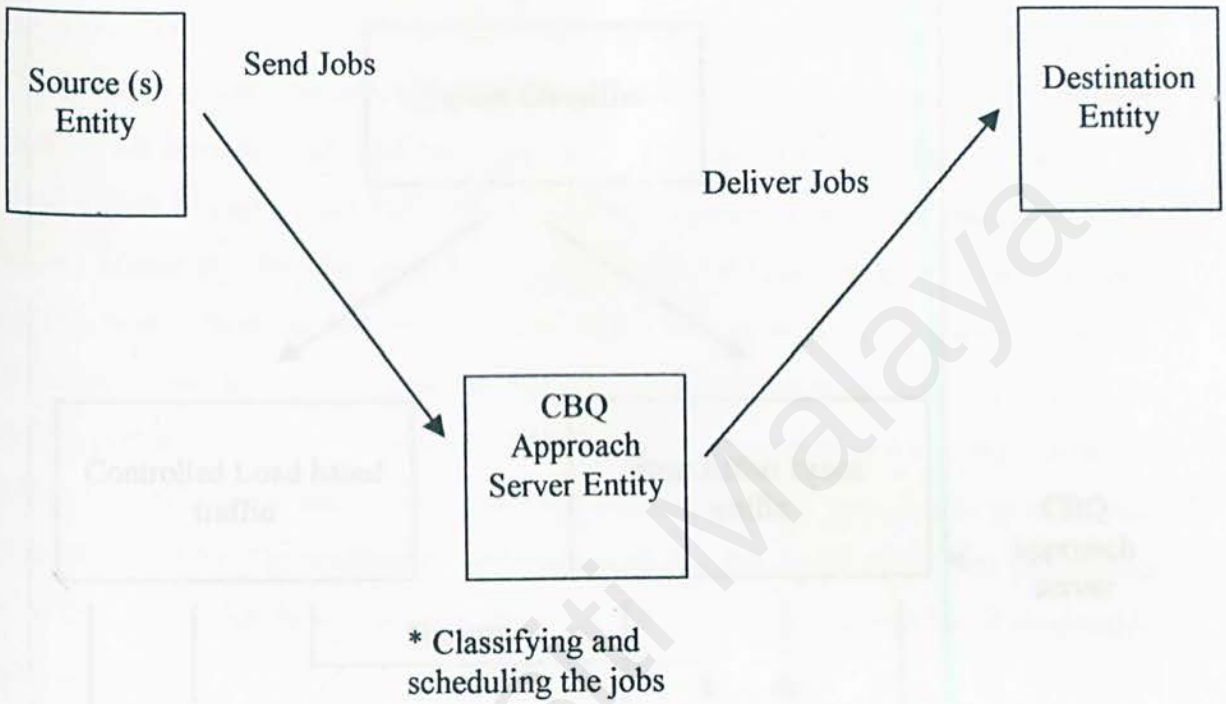
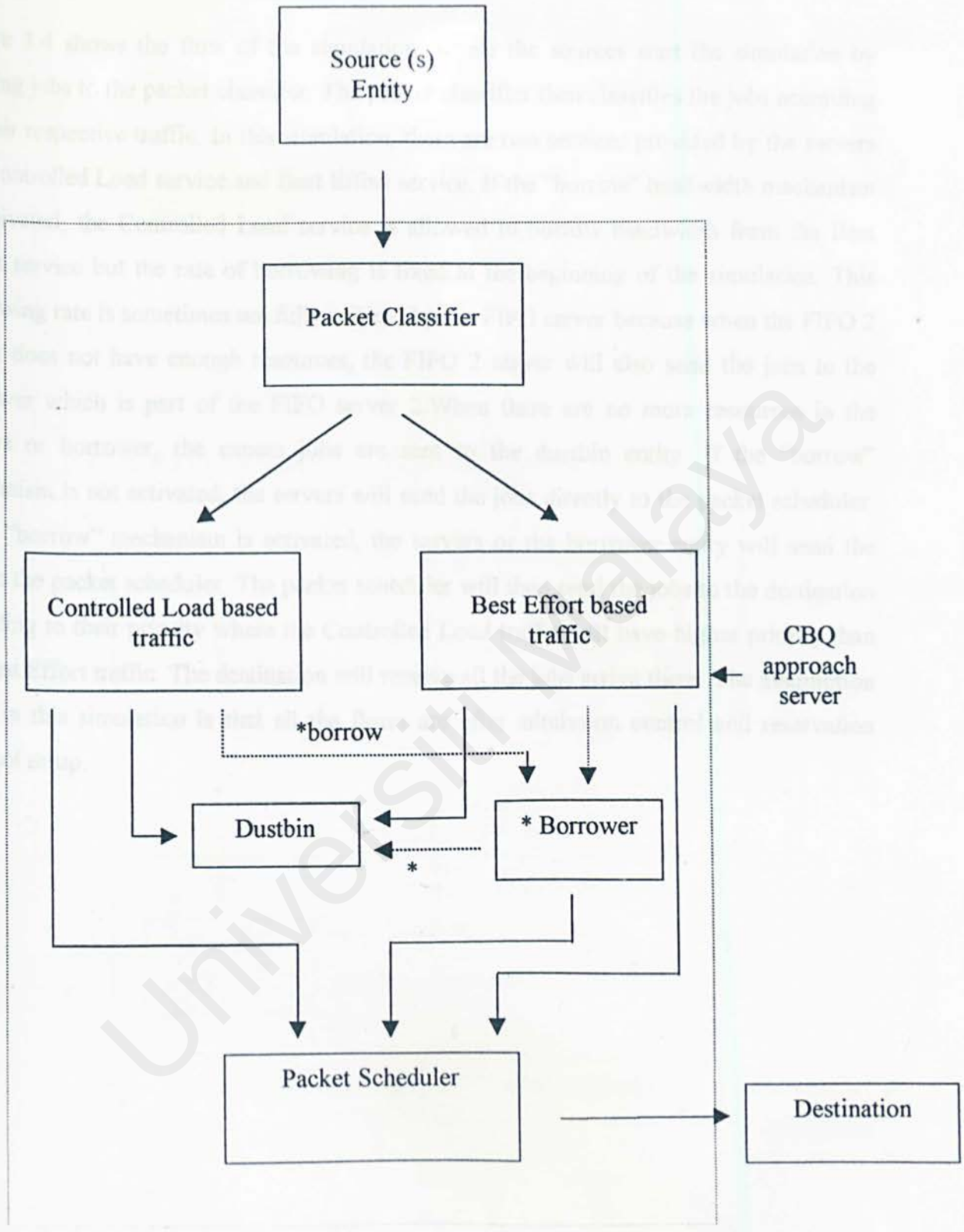


Figure 3.3 : General description about overall simulation

Note : This simulation assumes that the flow of traffic is after reservation protocol and admission control setup

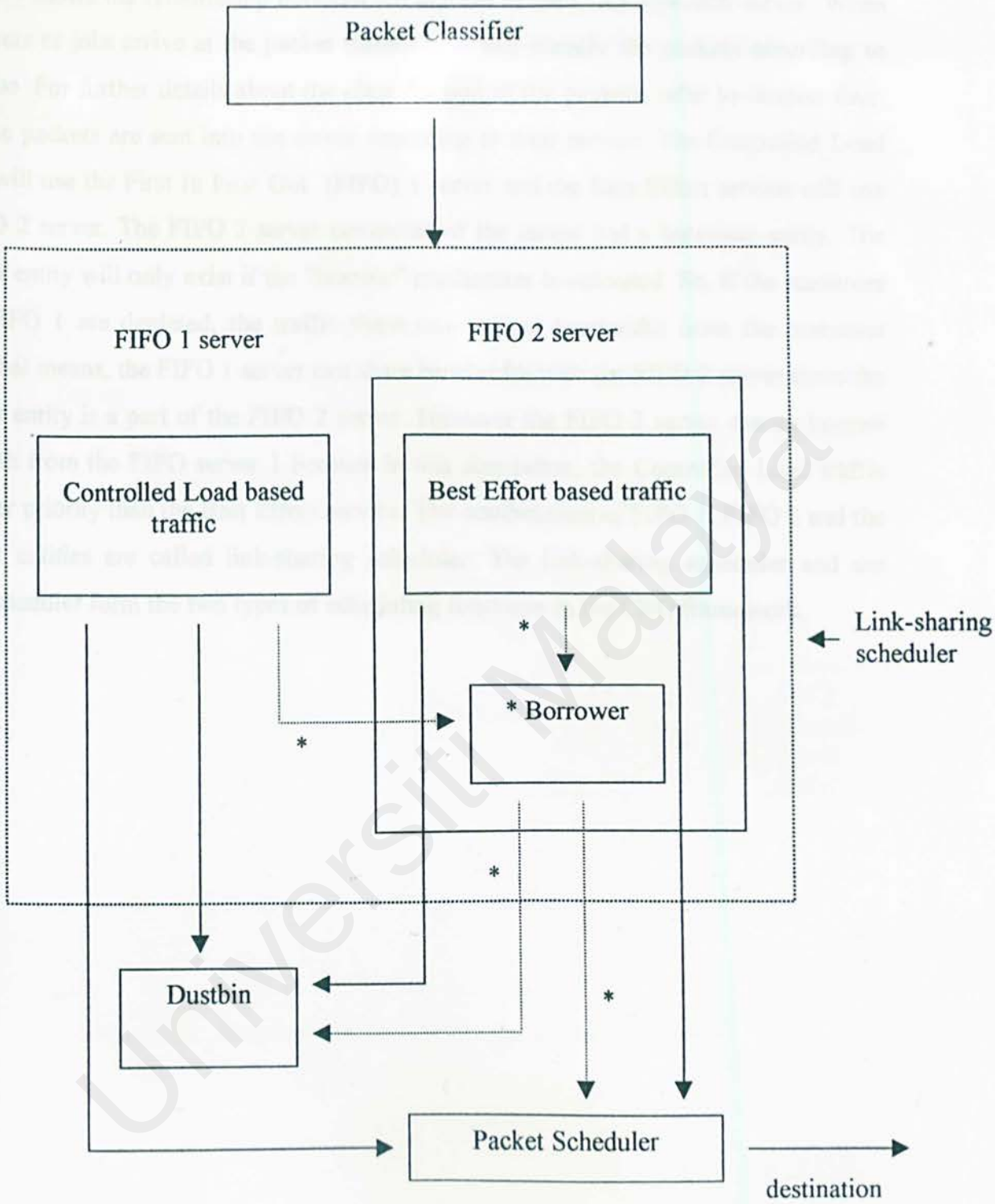


Note : * if activated

Figure 3 4 : More detail description about the simulation

3.3.1 The Flow Of The Simulation

Figure 3.4 shows the flow of the simulation, where the sources start the simulation by sending jobs to the packet classifier. The packet classifier then classifies the jobs according to their respective traffic. In this simulation, there are two services provided by the servers i.e. Controlled Load service and Best Effort service. If the "borrow" bandwidth mechanism is activated, the Controlled Load service is allowed to borrow bandwidth from the Best Effort service but the rate of borrowing is fixed at the beginning of the simulation. This borrowing rate is sometimes not fully utilized by the FIFO server because when the FIFO 2 server does not have enough resources, the FIFO 2 server will also send the jobs to the borrower which is part of the FIFO server 2.When there are no more resources in the servers or borrower, the excess jobs are sent to the dustbin entity. If the "borrow" mechanism is not activated, the servers will send the jobs directly to the packet scheduler. If the "borrow" mechanism is activated, the servers or the borrower entity will send the jobs to the packet scheduler. The packet scheduler will then send the jobs to the destination according to their priority where the Controlled Load traffic will have higher priority than the Best Effort traffic. The destination will receive all the jobs arrive there. The assumption made in this simulation is that all the flows are after admission control and reservation protocol setup.



Note : * if activated

Figure 3.5 : The entities in the CBQ approach server

Figure 3.5 shows the relationship between the entities in the CBQ approach server. When the packets or jobs arrive at the packet classifier, it will classify the packets according to their class. For further details about the classification of the packets, refer to chapter four. Then, the packets are sent into the server according to their service. The Controlled Load service will use the First In First Out (FIFO) 1 server and the Best Effort service will use the FIFO 2 server. The FIFO 2 server composes of the server and a borrower entity. The borrower entity will only exist if the "borrow" mechanism is activated. So, if the resources in the FIFO 1 are depleted, the traffic there can borrow bandwidth from the borrower entity. That means, the FIFO 1 server can share bandwidth with the FIFO 2 server since the borrower entity is a part of the FIFO 2 server. However the FIFO 2 server cannot borrow bandwidth from the FIFO server 1 because in this simulation, the Controlled Load traffic has higher priority than the Best Effort service. The combination of FIFO 1, FIFO 2 and the borrower entities are called link-sharing scheduler. The link-sharing scheduler and the packet scheduler form the two types of scheduling functions in the CBQ framework.

3.3.2 Creation Of Entities

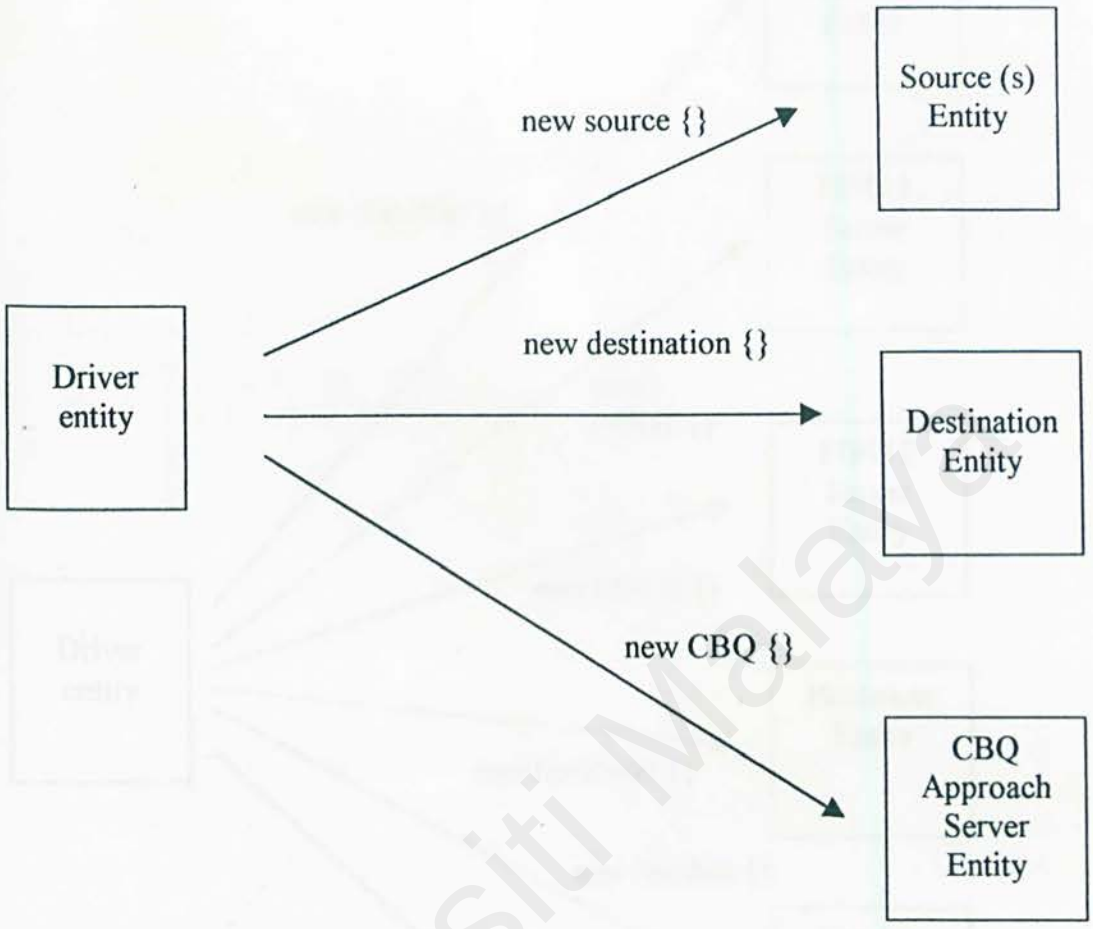


Figure 3.6 : Creation of the entities using the new command

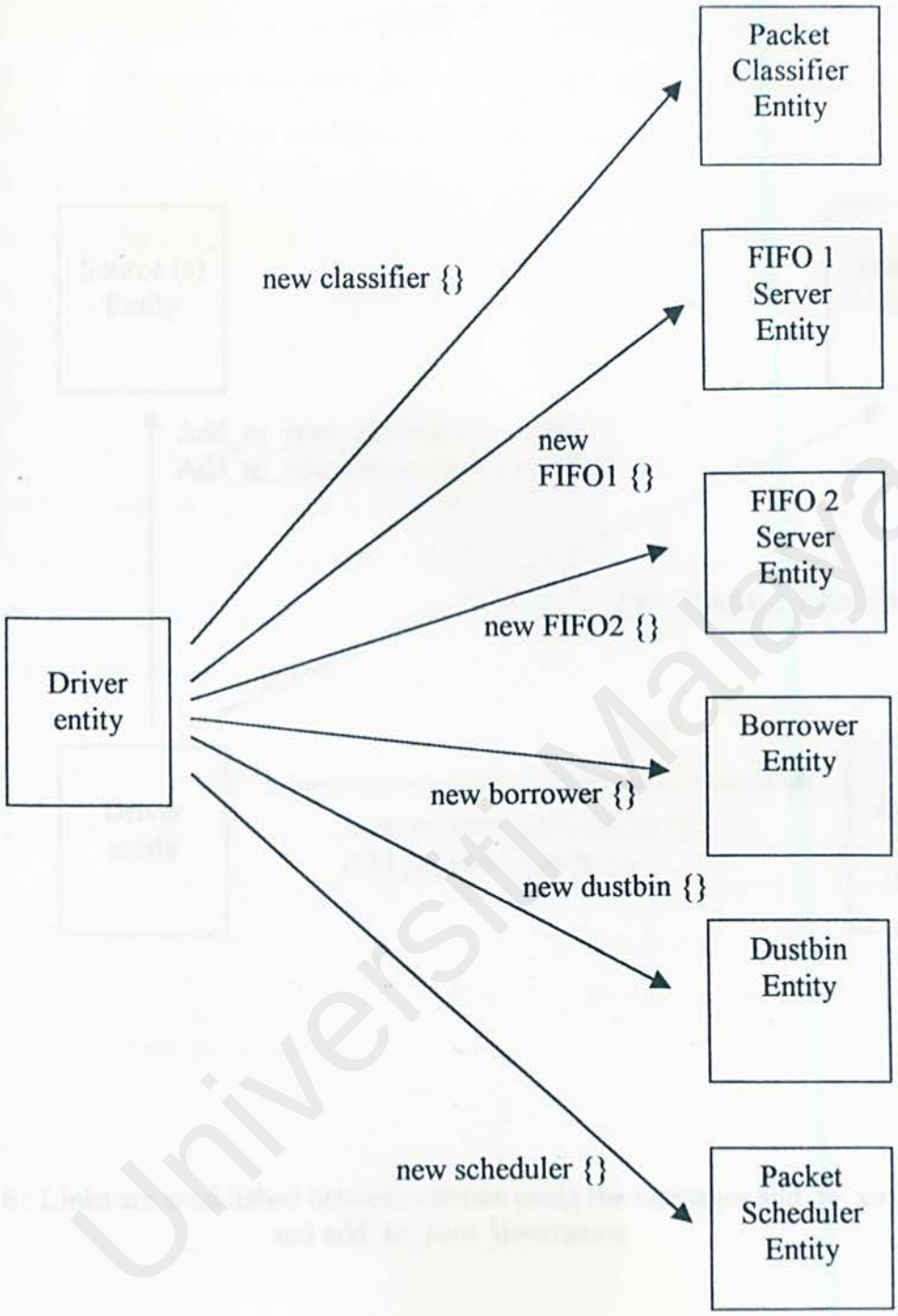


Figure 3.7 : Creation of the entities in the CBQ approach server using the new command

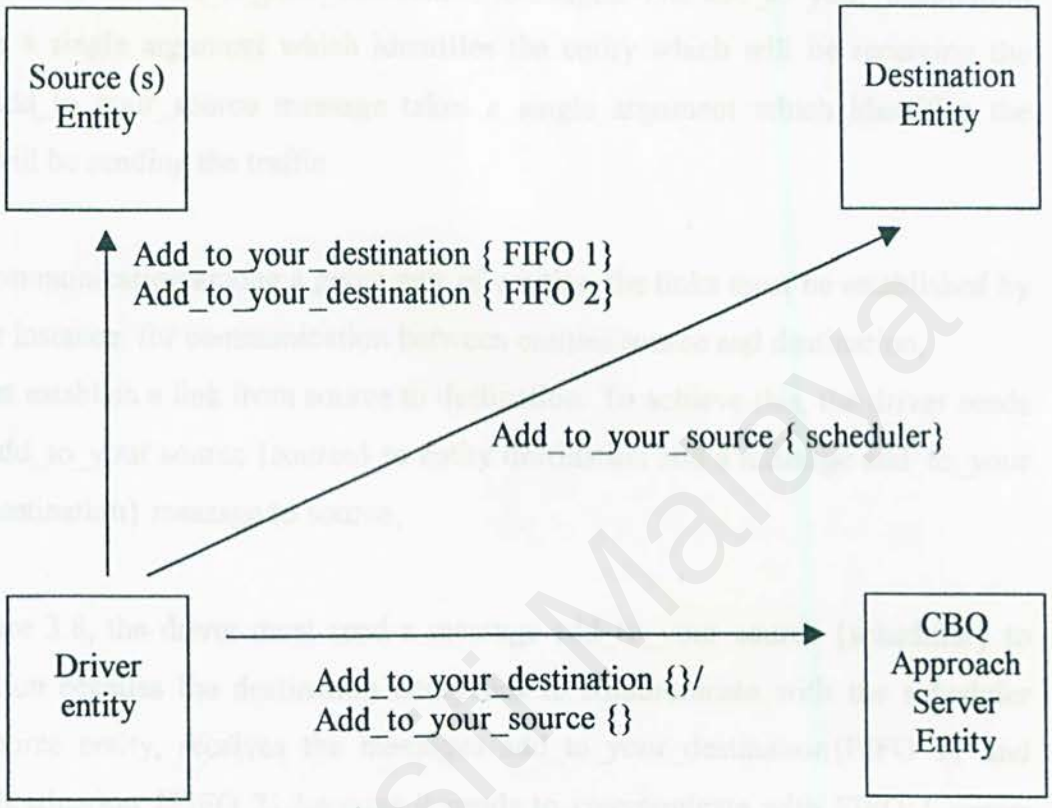


Figure 3.8 : Links are established between entities using the messages add_to_your_source and add_to_your_destination

3.3.3 Links Establishment

Figure 3.8 shows the second step after the creation of new entities by the driver entity. In the second phase, links are established between the entities using the messages add_to_your_source and add_to_your_destination messages. The add_to_your_destination message takes a single argument which identifies the entity which will be receiving the traffic. The add_to_your_source message takes a single argument which identifies the entity which will be sending the traffic.

Prior to any communication among a given pair of entities, the links must be established by the driver. For instance, for communication between entities source and destination, the driver must establish a link from source to destination. To achieve this, the driver sends the message add_to_your source {source} to entity destination and a message add_to_your destination {destination} message to source.

Hence, in figure 3.8, the driver must send a message add_to_your source {scheduler} to entity destination because the destination entity has to communicate with the scheduler entity. The source entity, receives the messages add_to_your_destination{FIFO 1} and add_to_your_destination {FIFO 2} because it needs to communicate with FIFO 1 server and FIFO 2 server. Refer to Figure 3.4 for further information about the links and communication between the entities.

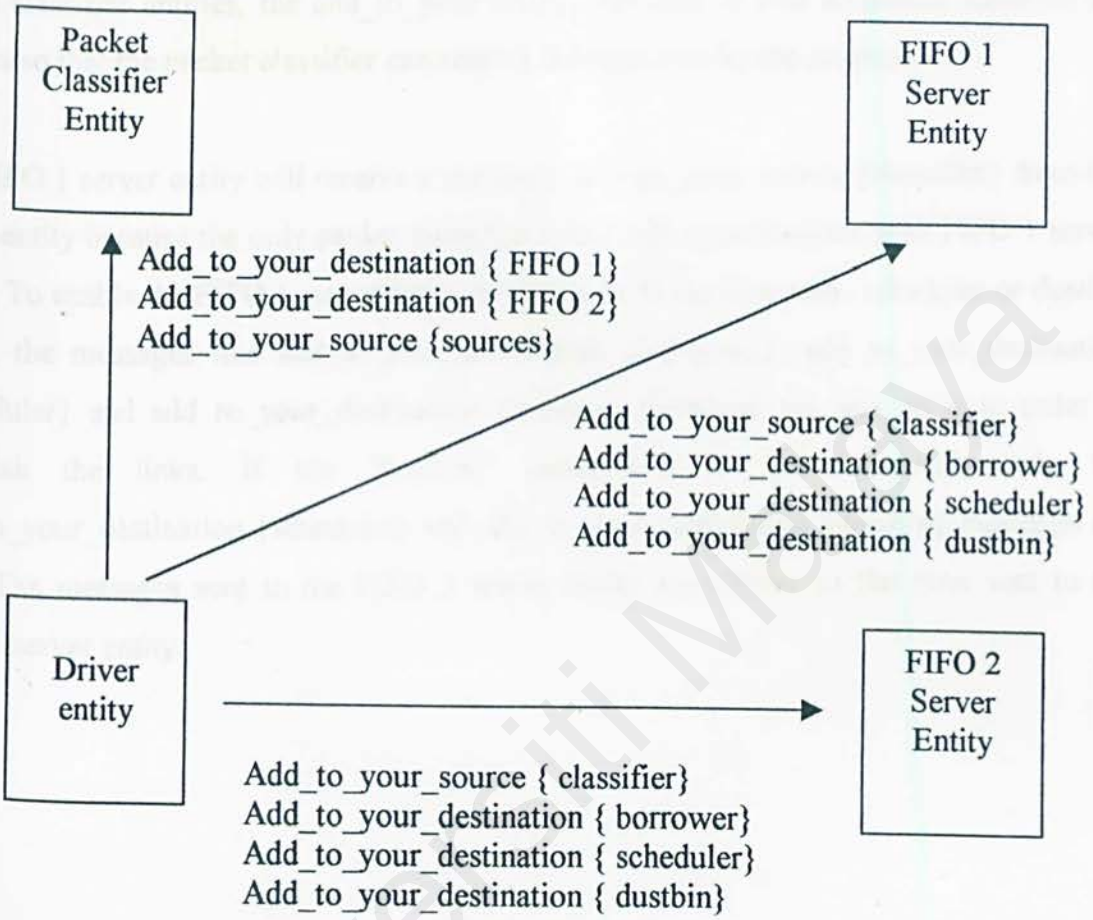


Figure 3.9 : Links are established between entities in CBQ approach server using the messages add_to_your_source and add_to_your_destination

In figure 3.9, the packet classifier entity will receive messages add_to_your_destination {FIFO 1} and add_to_your_destination {FIFO 2} because after classifying the packets according to their respective traffic or server, the scheduler entity has to send the packets either to FIFO 1 or FIFO 2 servers. In order to establish the links between the sources and packet classifier entities, the add_to_your source {source} is sent to packet classifier by sources so that the packet classifier can receive the jobs sent by the sources.

The FIFO 1 server entity will receive a message add_to_your_source {classifier} from the driver entity because the only packet classifier entity will communicate with FIFO 1 server entity. To enable the FIFO 1 server entity to send jobs to the borrower, scheduler or dustbin entity, the messages like add_to_your_destination {borrower}, add_to_your_destination {scheduler} and add_to_your_destination {dustbin} messages are sent to it in order to establish the links. If the "borrow" mechanism is not activated, only the add_to_your_destination {scheduler} and add_to_your_destination {dustbin} messages are sent. The messages sent to the FIFO 2 server entity are similar to the ones sent to the FIFO1 server entity.

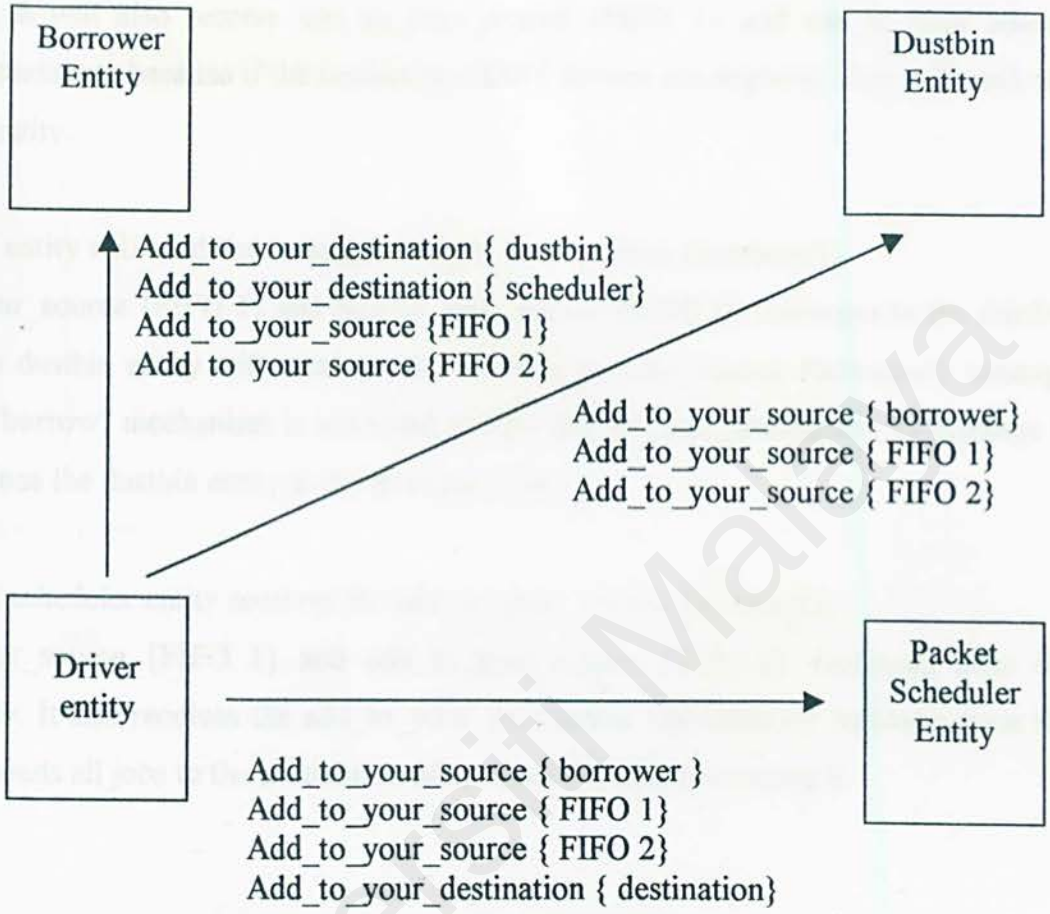


Figure 3.10 : Links are established between entities in CBQ approach server using the messages add_to_your_source and add_to_your_destination message

In figure 3.10, when the borrower entity is activated, it will receive the add_to_your_destination {dustbin} and add_to_your_destination {scheduler} messages because it needs to send jobs to the dustbin entity if all resources are fully utilized. On the other hand it will send the jobs to the packet scheduler if enough resources are available. Moreover, it will also receive add_to_your_source {FIFO 1} and add_to_your_source {FIFO 2} messages because if the resources at the 2 servers are depleted, they will seek the borrower entity.

The driver entity will send the messages add_to_your_source {borrower}, add_to_your_source {FIFO 1} and add_to_your_source {FIFO 2} messages to the dustbin entity. The dustbin entity will receive only the add_to_your_source {borrower} message when the "borrow" mechanism is activated and no add_to_your_destination {} message is received since the dustbin entity is the terminal entity.

The packet scheduler entity receives the add_to_your_source {borrower}, add_to_your_source {FIFO 1} and add_to_your_source {FIFO 2} messages from the driver entity. It also receives the add_to_your_destination {destination} message since the scheduler sends all jobs to the destination after it has finished processing it.

3.3.4 Topology Establishment

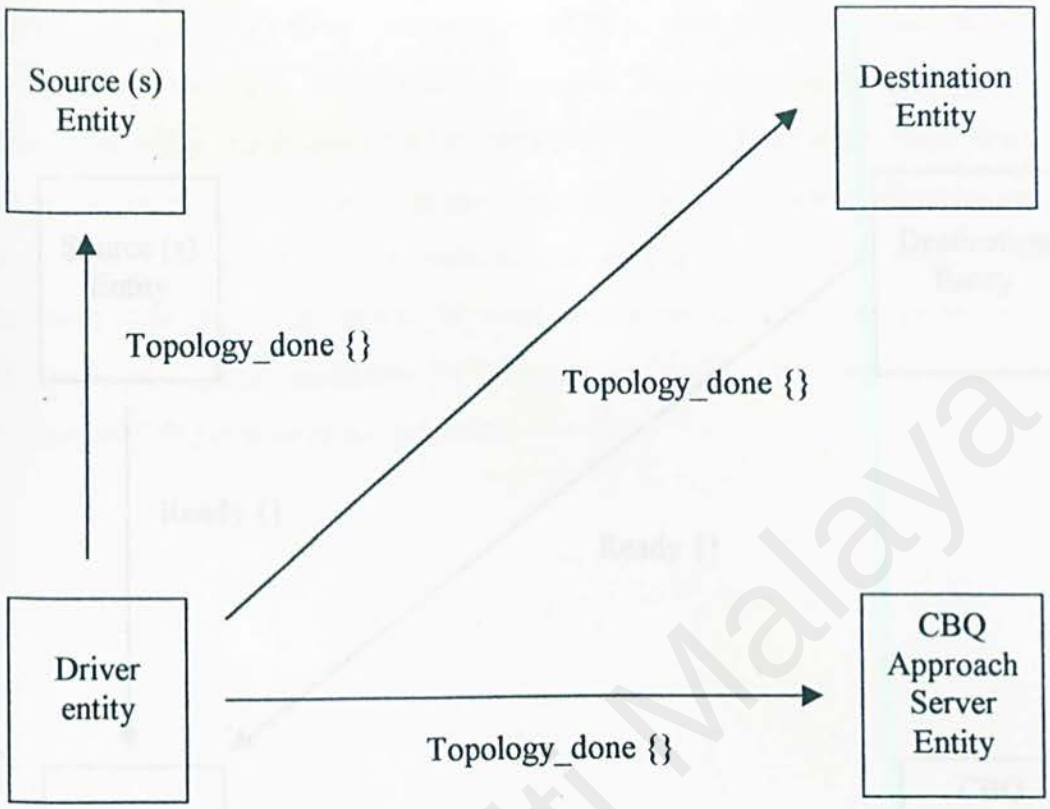


Figure 3.11 : The driver sends message topology_done to all the entities

In the third phase, after the complete topology has been specified, the driver entity sends the topology_done{} message to all the entities in the model. In figure 3.11, the driver entity sends six topology_done {} messages to the CBQ approach server entity since it contains six entities.

3.3.5 Handshake Process

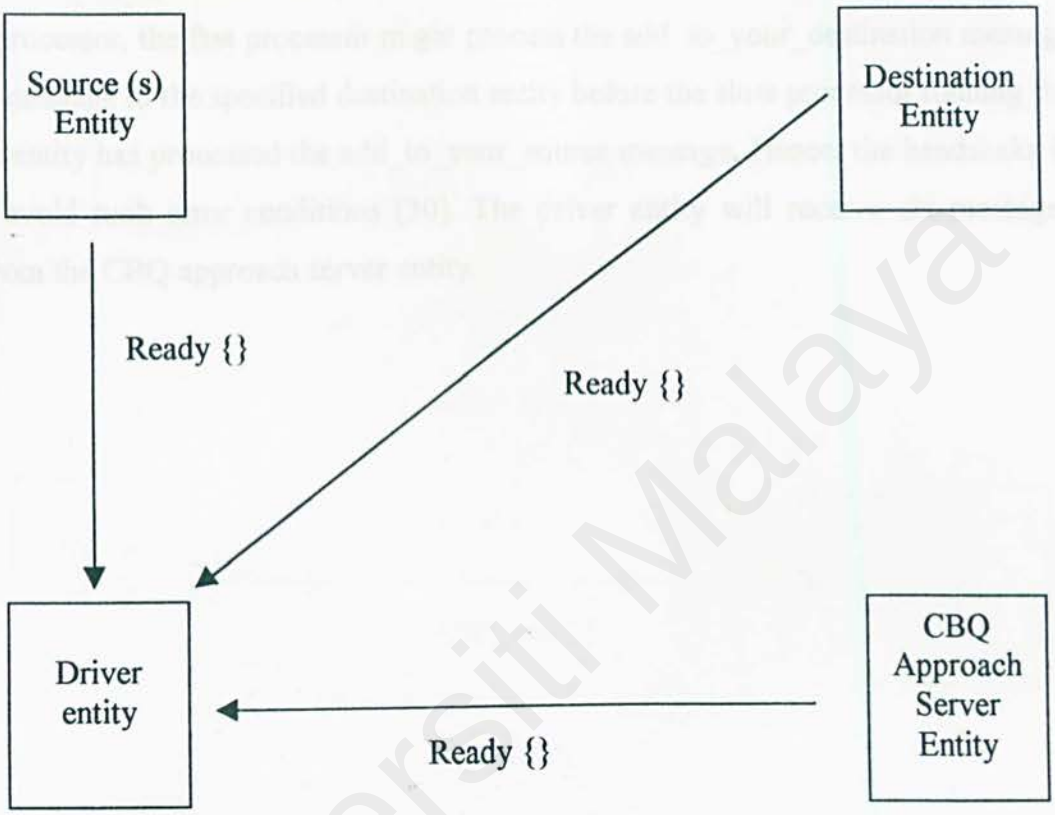


Figure 3.12 : All entities send a ready message to the driver

After processing their topology establishment messages, all entities return a ready message to the driver. The reception of this message from all entities in the network tells the driver that the topology has been successfully established. Note that the handshake is needed at the end to account for slow processing of the add_to_your _destination and add_to_your_source messages at one of the processors. If we omit this handshake, in a case where the source entity is running on a fast process while the destination entity is running on a slow processor, the fast processor might process the add_to_your_destination message and send a message to the specified destination entity before the slow processor running the destination entity has processed the add_to_your_source message. Hence, the handshake is needed to avoid such error conditions [30]. The driver entity will receive six messages ready {} from the CBQ approach server entity.

3.3.6 Simulation Starts

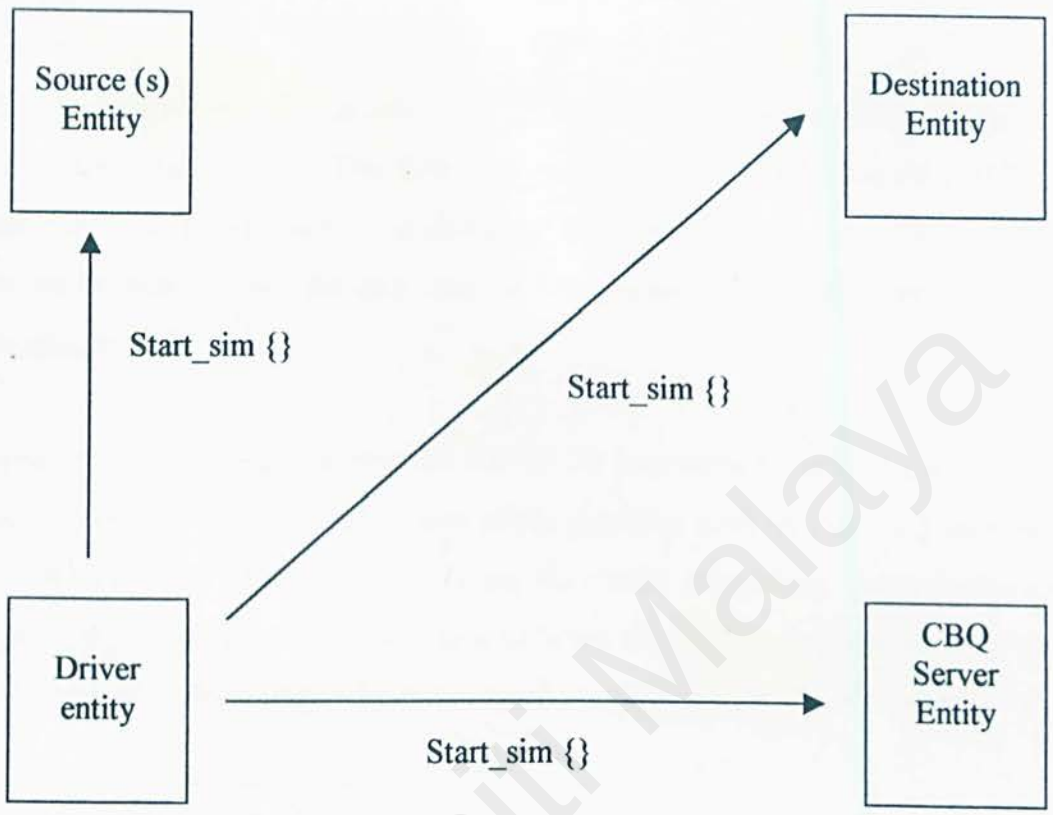


Figure 3.13 : Driver sends start_sim message to the entities and the simulation starts

In the final phase, on receiving ready messages from all entities, the driver sends start_sim messages to the entities and the simulation starts.

Chapter 4

Implementation Of The Traffic Scheduling And Link Sharing Mechanism Simulation

4.0 Introduction

The simulation architecture for this project is based on a modular design which divides the simulator into three mains parts. The first part is the Class Based Queueing (CBQ) approach server and the second part is the simulator sources and destination. The last part is the driver entity which plays the key role in every simulation written by PARSEC simulation language.

We have chosen the modular design because one of the important benefit of the modular design is that is relatively easy to replace any of the modules such as the CBQ approach server entity. Besides that, the entities simulating the traffic scheduling and link-sharing mechanism process can be easily integrated in a different simulation architecture. It is also easy to rectify problems in the modular design and thus enhancing the efficiency of the simulator.

The rest of this chapter is structured as follows: In section 4.1 we talk about the simulator architecture and the overall design overview. Section 4.2, we focus on the entity scheduling, and in the following section we discuss the network traffic flow. Section 4.4, we present the implementation of the Class Based Queuing (CBQ) approach server. This is followed by the implementation of each entity in the CBQ approach server starting with the classifier entity, then the FIFO servers entities, the borrower entity, the dustbin entity and the last entity is the scheduler entity. In section 4.5, we talk about the implementation of the sources and the following section discuss the calculation of the processing and hold time for a packet. Section 4.6 brings on the implementation of the destination, and the last section focus on the compiling of this simulation program and also some brief description about the files in this simulation.

4.1 Simulator Architecture

As described above, the simulator is divided into 3 parts (see figure 4.1 for more details.) The Class Based Queueing (CBQ) approach server is responsible for scheduling the packet traffic and maintaining the link-sharing constraint among the classes. There are five entities inside the CBQ approach server and the function of each entities have been discussed in detail in chapter three.

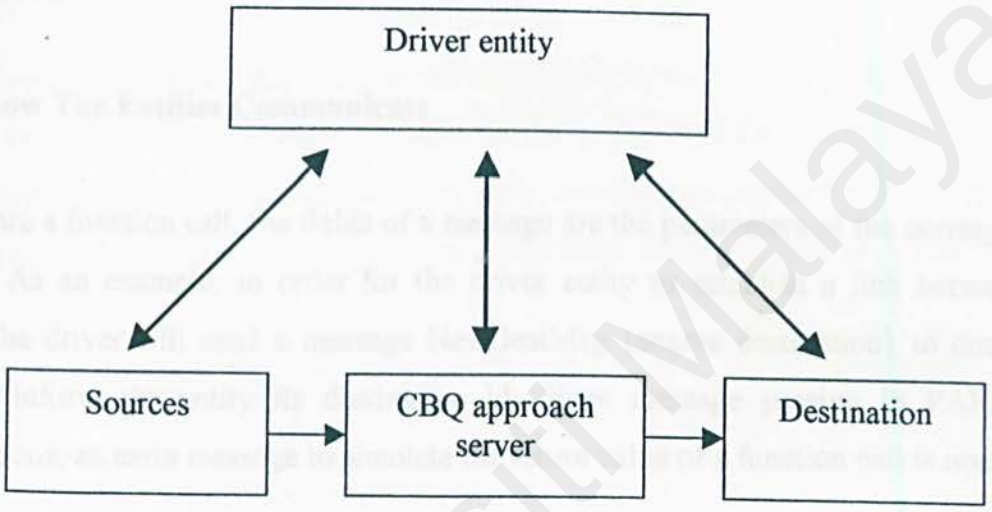


Figure 4.1 The Simulator Architecture

The simulator engine is composed of a driver entity which is responsible for creating the rest of the entities. It is also responsible for communicating with the sources or destination, with the CBQ approach server as well as the entities that implement the network layer. Every PARSEC program must include the driver entity which serves a purpose similar to the main function of a C program. Execution of a PARSEC program is initiated by executing the first statement in the body of entity driver.

The last part of the simulator is the source and the destination entities. The role of the sources is to initiate jobs or packets while the destination receives the jobs or packets accordingly to their respective classes. We discuss the sources and destination in more

details in the next section.

4.1.1 An Overview Of The Entities Communication

In this project, we use message passing to simulate function call and packet delivery since PARSEC provides powerful message receiving constructs that result in shorter and more natural simulation programs. Entities communicate with each other through buffered message passing. A unique message buffer is associated with each entity with size of 240 KB by default.

4.1.1.1 How The Entities Communicate

To simulate a function call, the fields of a message are the parameters of the corresponding function. As an example, in order for the driver entity to establish a link between two entities, the driver will send a message NewDestMsg {ename destination} to one of the entity to inform the entity its destination id. Since message passing in PARSEC is asynchronous, an extra message to simulate the return value of a function call is needed.

4.1.1.2 How To Simulate Packet Delivery

In order to deliver a packet to a destination, a source entity sends a message that contains the packet's intended destination. The message may travel through many entities before it reaches its destination. The receiving entity need not unpack the message to retrieve the packet since in this simulation, we use a flow-id field in the message to reduce the overhead of the process. It also allows a short cut classification of the packets.

The receiving entity can also unpack the message to retrieve the packet and confirm its destination by checking the parameter list in the message, doing so will increase the overall overhead of the simulation. If the receiving entity is not the intended destination of the packet, it may again forward the packet by sending a message.

4.1.2 How The Entities Know Each Other

Each entity needs to know the entity id or ename of other entities in order to communicate with them. As discussed in chapter 3, an ename is a new type introduced by PARSEC to store the unique identifier given to an entity. It is important to make sure that the communicating parties is in the right order just for the simulation to run. If links are not well established between entities, a core dumped error will appear. For example, the classifier entity must know the First In First Out server entity to be able to forward packets. It may also need to know the sources entity to receive packets.

Hence, a well established link between all the entities is very important and the success or failures of the simulation depends heavily on the links. Here the driver entity comes into the picture and its main responsibility is to distribute the communication topology (not the network topology) to other entities after creating them. Each entity is identified through its ename.

4.2 Entity Scheduling

Before we proceed to the implementation of each part, we must know the scheduling of an entity to understand the implementation of each entity. In a PARSEC program, an arbitrary number of entities may be mapped to a single processor. The execution of these entities is interleaved by the PARSEC scheduler. Entities are scheduled for execution based on the timestamps of their enabling messages [8].

An entity have four types of states and can be in one of the four states. Firstly is the terminated state where an entity that has been terminated does not participate any further in the program. The second state is the idle state where an entity that has not been terminated is said to be idle. It is in idle states when its message buffer does not contain any enabling message. When the buffer contains an enabling message, the entity is in ready state and at any given point, multiple entities on a processor may be in the ready state. The scheduler

will select an entity in the ready state with the earliest enabling message for execution, and the selected entity enters the active state again.

An active entity relinquishes control to the scheduler only if it is terminated or it enters a hold or receive statement. When an active entity receives a message and the buffer contains an enabling message, it will transit to the ready state which makes it eligible to become active again. But if the buffer does not contain any enabling message, it will transit to the idle state. The important thing here is that the scheduler cannot force an entity to relinquish control and an active entity is self-scheduled. An active entity will not relinquish control to the scheduler if it never executes a receive or hold statement.

4.3 Network Traffic

In this simulation, JobMsg{} is used to simulate packets that act as traffic flow across the network link. The syntax for the JobMsg{} message is as follows :

JobMsg{**char** number, **int** TOS}

The JobMsg message contains of two parameters which are the number and type of service (TOS). The message carries the information about the number of packets that are sent across the network link. The TOS parameter enables the packet classifier to classify the packets according to its required service. The packet scheduler may use this parameter to identify which class of service the message would receive and allows higher priority packets to be sent first.

JobMsg{} is assumed to represent a fixed length packet which is equivalent to the token length of 1000 bits. Thus, the message parameters do not represent any information about the packet length or size. Besides that, this message also omits all the IP header information that is not necessary. The assumption of JobMsg is a fixed length packet enables different length to be used without any changes in the simulation design. The use

of fixed length packet also enhance and simplify the overall measurement process at the FIFO server entity. The flow of JobMsg is shown in Figure 3.14.

4.4 Implementation Of The CBQ Approach Server

There are five entities in the CBQ approach server: a link-sharing scheduler which consists of two First In First Out (FIFO) server, a classifier, a borrower, a dustbin and a packet scheduler. We discuss each entity implementation in the following sections.

4.4.1 Implementation Of The Classifier Entity

As discussed in Chapter 3, the role of a classifier is to classify the packets to their respective classes by looking at its flow-id, source or destination address, protocol number etc. The sources send messages to the classifier and when the classifier receives a message, it will classify the message according to its flow-id. For example, when the classifier receives a packet which contains JobMsgB {char number, int TOS}, it will automatically forward the message to the FIFO server two for further processing because the flow-id of this packet is B (the B comes from the flow-id of the JobMsgB{}). Another way for the classifier to know which class the message belongs to is to unpack the message. After that, the classifier will check the message parameter Type Of Service (TOS). If it is set to 1, it forwards the message to FIFO server 1 for Best Effort service whereas if it is set to 0, it forwards the message to FIFO server 2 for Controlled Load service. However, the latter increases the overhead of the flow because it takes time to unpack the message and resends it.

4.4.2 Implementation Of The FIFO Server Entity

A FIFO server entity has the classifier entity as its source while its destination are the borrower entity (if the borrow mechanism is activated) and the scheduler entity and the dustbin entity (if the borrow mechanism is not activated). In every FIFO server entity, there

are a few tokens which represent the bandwidth allocated to that server. The calculation of the bandwidth is discussed later in the next section. The FIFO server 1 is for the Controlled Load service while the FIFO server two is for the Best Effort service. When the FIFO server one do not has enough tokens (bandwidth), it will send subsequent messages to the borrow entity if the borrow mechanisms is activated. The same happens for FIFO server 2 but the FIFO server 2 has a higher priority than the FIFO server 1 in using the tokens in the borrow entity because the borrow entity is part of the FIFO server 2.

If messages from the FIFO server 1 are consuming the tokens and congestion occurs, the FIFO server 2 will send an Overlimit {} message to the borrower entity. The function of the Overlimit{} message is to alert the borrower entity to preempt all messages in the buffer so that the packets from the FIFO server 2 can consume the tokens. The packets from the FIFO server 1 can continue their work after the packets from the FIFO server 2 have finished using it. This is the unique features or constraints of link-sharing mechanism which states that when the borrower needs their allocated bandwidth back, the class which has borrowed from it must return it.

The big question here is that how the borrower entity knows when the FIFO server 2 messages has finished using the tokens when the congestion occurred so that the messages from FIFO server one can continue to use the tokens from the borrower entity. To overcome this problem, we use the timeout clause. PARSEC supports both timeout-first and timeout-last semantics. In this simulation, we chose the timeout last semantics which makes use of the **after** keywords because it is more powerful and easy to use. So, when the borrower entity didn't receive any messages from the FIFO server 2 in a particular time, it will timeout and the packets from the FIFO server 1 may continue consuming tokens at the FIFO server 2.

Each message which contains a packet or a job must be held for a certain period of time in the FIFO servers where the period according to the link bandwidth. If the borrow mechanism is activated, all messages which contains packets or jobs must be through the borrower entity before it reach another destination. On the other hand, if the borrower

mechanism is not activated, the FIFO servers will send the messages directly to the scheduler entity after processing it, or to the dustbin entity if congestion occurs or if tokens are depleted.

4.4.3 Implementation Of The Borrower Entity

If the borrow mechanism is activated, the borrower entity may have the FIFO server entities as the source and the scheduler and dustbin entities as the destination. The function of the borrower entity is to allow the FIFO server 1 to borrow resources or bandwidth at a fix rate from the FIFO server 2. The borrower entity is a part of the FIFO server 2. When the tokens inside the borrower entity is depleted, all subsequent packets will be sent to the dustbin entity. The borrower entity gives a higher priority to the FIFO server 2 entity and will only serve the FIFO server 1 messages when FIFO server 2 is not fully utilized or there is no congestion.

4.4.4 Implementation Of The Dustbin Entity

The function of the dustbin entity is to accumulate and store all excess packets or unwanted packets. The dustbin entity has the borrower entity (if borrow mechanism is activated) as its source or FIFO servers entities if borrow mechanism is not activated as its source. It does not have any destination entity(s) because it is a terminal entity. The dustbin entity will accumulate all packets according to their classes and at the end of the simulation, it prints out all the accumulated statistical data.

4.4.5 Implementation Of The Scheduler Entity

The scheduler entity is the last entity in the CBQ approach server and it plays a major role in scheduling traffic by sending packets from the highest priority level first. In this simulation, the scheduler entity has the FIFO servers and borrower entities as the source and after processing the packets, it will deliver them to the destination.

We use the priority based scheduling algorithm which means packets from the highest priority level are sent first. In this case, the Controlled Load service obtains a higher priority than the Best Effort service. The scheduler entity uses one of the feature of advanced message receive constructs which is the Qempty (m_t) command. This function returns true only if the buffer does not contain any m_t messages. It will return false if the buffer has that message. For instance the following receive statement gives higher priority to the Overlimit message. It receives JobMsgA only when no Overlimit message is available.

```
receive ( JobMsgA jobA ) when  ( qempty ( OverlimitMsg ) )
{
        .....
}
or receive ( Overlimit ovr )
{
        .....
}
```

4.5 Implementation Of The Sources

The role of the sources or senders is to generate packets and to send it to the next entity. Thus, the sources initiate the whole link-sharing and traffic scheduling mechanism procedure. A source entity generates the packet and takes the type of service required by the packets and also the id of the packets as the parameters. The sources send the packets after a period time or in other words, the sources initiate the packets within a certain period. This allows other entities to have time to service a packet. In order to simulate an overload situation, the sources generate twice as many jobs that are generated during a normal situation. Similarly, to simulate a lightly loaded situation, the sources generate half the number of packets generated during the normal situation. The calculation of the time taken to process a packet and also the hold time for a packet are discussed below.

4.5.1 Calculation Of The Processing And Hold Time

In this simulation, we assume that a packet is 1000 bit long and the link bandwidth chosen is 3Mbps for the total of the two servers. The source is modeled as a Markov-modulated fluid process with the smallest unit is Bit[31]. Thus, to simplify the calculation and measurement in the FIFO server entity, 1000 bit is chosen. There are three different link bandwidth used in this simulation (ie 1.5, 3.0 and 10 Mbps) and all these three link bandwidth are chosen because they are commonly used in the traffic algorithm. Therefore in one second, the link can support for 3×10^6 bit and the processing time for one packet of 1000 bit is :

$$= (1000\ \text{b} \times 1\ \text{s}) \div (3 \times 10^6)$$

$$= 3.33 \times 10^{-4}\ \text{s}$$

$$= 333\ \mu\text{s}$$

Therefore, in this simulation one time unit is equal to 10 μs and the process or hold time for one packet is 333 time unit. Different link bandwidth uses different hold or process time for a packet, and a different percentage of bandwidth allocated to each class also uses different hold or process time.

4.6 Implementation Of The Destination

The main function of the destination or receiver is to collect all traffic and the traffic which has been collected is used to analyze the simulation performance. The destination will collect the packets according to their classes through the flow id or the type of service

provided in the message parameters. The destination entity has the scheduler entity as the source.

4.7 Description Of The Program Files And Compiling The Program

This section describes briefly the files in this simulation program and how the program is complied since this program requires a few additional options. There are 7 files in this program and below is the description of the files.

1. simnet.pc : this file contains the driver entity code, the sources entity code

2. classi.pc : this file contains the classifier entity code

3. FIFO.pc : this file contains the FIFO entity code

4. extra.pc : this file contains the borrower entity code and also the dustbin entity code.

5. scheduler.pc : this file contains the scheduler entity code

6. dest.pc : this file has the destination entity code

7. servers.h : this is the header file for the servers.

The PARSEC compiler called pcc, accepts all the options supported by the C compiler. It also supports separate compilation. PARSEC programs are usually given a .pc extension and the compiler also creates a .pi file for each source file. The .pi file contains information about the message types used in the file and must be visible to the compiler at the link stage. It is excluded if all the messages types in it are also used in other files.

To compile this program, we need to use the pcc_linker option to link all the separate files together and the full command is :

pcc –pcc_linker gcc simnet.pc classi.pc FIFO.pc extra.pc scheduler.pc dest.pc –lm

This generates an executable file **a.out** in the current working directory.

Chapter 5

Testing And Results

5.0 Introduction

The purpose of the testing procedure was to compare the effectiveness of the "borrow" mechanism, which is one of the main feature of the link-sharing mechanism under different link bandwidth or even in different situation during congestion period. The result of the test can determine of the algorithm to provide bandwidth guarantee to the higher priority classes. Hence, this simulation test was divide into two main parts, where one test used the borrow mechanism while another test is without the borrow mechanism.

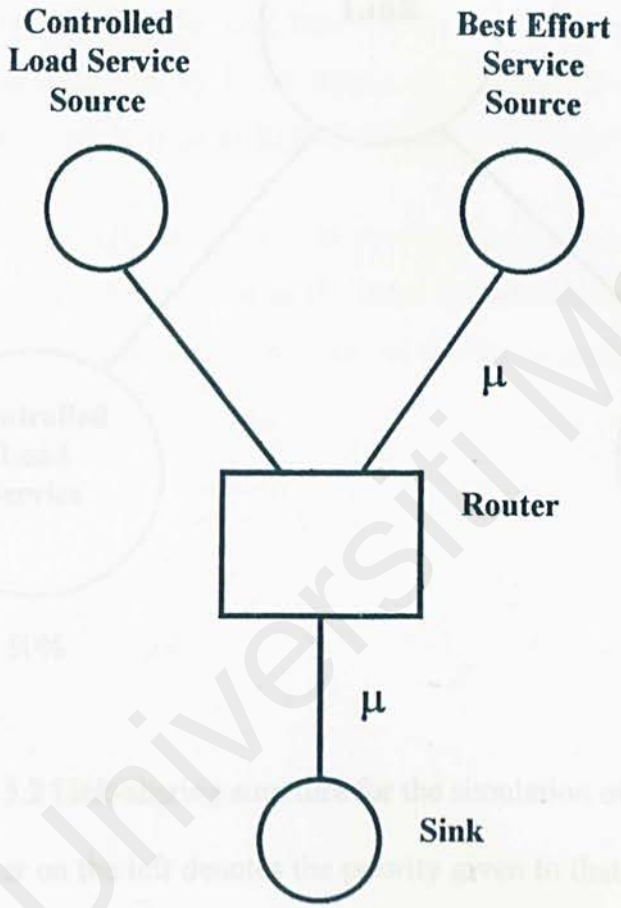
This simulation has made an assumption that all the flows or traffics are accepted after the admission control procedure or any resource reservation setup. This is because the objective of this project is to observe the efficiency of the link-sharing mechanism. In this project, we also did not look into the delay jitter experienced by the packets because of its complexity.

The rest of the chapter is organized as follows. In Section 5.1, we describe the simulation scenario and in the following section, we investigate the types of test to be simulated in order to determine the efficiency of the borrow mechanism. In Section 5.1.3, we talk about the test output parameters. We analyze the result of different types of test in Section 5.2, and finally the discussion is in the last section.

5.1 Testing

5.1.1 Description Of The Simulation Scenario

In the development and testing of this simulation implementation, the network scenario below is used.



where μ is the link bandwidth or capacity and depends on simulation situation

Figure 5.1 Network Scenario For The Simulation

The router contains the Class Based Queueing (CBQ) server approach to execute the link-sharing and traffic scheduling mechanism. For this project, the simulation was built and

compiled using the Solaris 2.5.1 version of PARSEC compiler. We used the SunOS 5.7 as the operating system for compiling and executing the simulation.

In this project, the simulator use the flat link-sharing structure for all tests (see figure below for more detail).

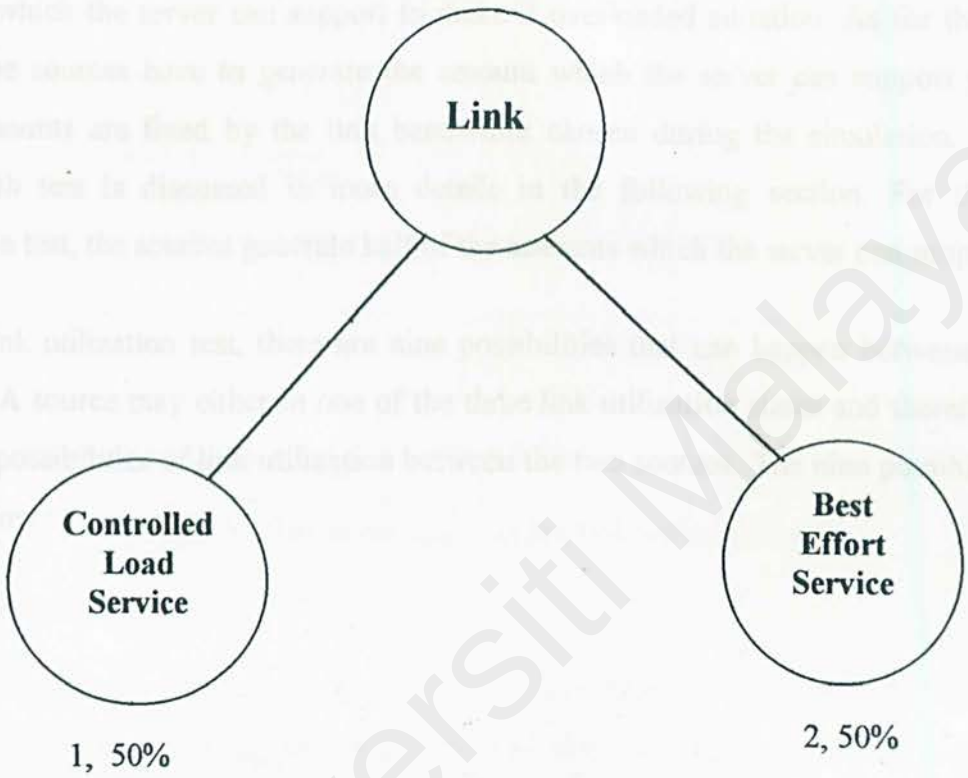


Figure 5.2 Link-sharing structure for the simulation of flat link-sharing

Note : The number on the left denotes the priority given to that class by the root class and the number on the right denotes the percentage allocates to that class. The percentage above is merely an example and varies in each test

5.1.2 Types Of Test

As mentioned earlier, all tests are divided into two parts: one part with the borrow mechanism activates and another part without the borrow mechanism. These two types of test were done under different link utilization and different link bandwidth

5.1.2.1 Link Utilization

There are three types of link utilization: overload, normal and lightly utilized .The overload test represents a congestion period while the normal test represents a heavy utilization of the link. Last but no least, the lightly utilized state automatically represents the normal utilization of the link bandwidth. Since, the packet size is fixed and a packet size is equivalent to 1000 bits, the sources have to generate twice the amount of the packets which the server can support to make it overloaded situation. As for the normal mode, the sources have to generate the amount which the server can support only. All these amounts are fixed by the link bandwidth chosen during the simulation. The link bandwidth test is discussed in more details in the following section. For the lighty utilization test, the sources generate half of the amounts which the server can support.

In this link utilization test, there are nine possibilities that can happen between the two sources. A source may either in one of the three link utilization states and therefore there are nine possibilities of link utilization between the two sources. The nine possibilities are listed below :

	Controlled Load Service Source	Best Effort Service Source
1.	Normal	Normal
2.	Light	Light
3.	Normal	Light
4.	Light	Normal
5.	Over	Light
6.	Normal	Over
7.	Light	Over
8.	Over	Normal
9.	Over	Over

Over = overload, Normal = normal and Light = lightly

Tables 5.1 The possibilities in the link utilization test

5.1.2.2 Borrowing rate

If the borrow mechanism is activated, there are three types of rate which are tested. The three types of rate are 10%, 20% and 30%. The important thing is that the rate here is fixed at the beginning of the simulation and the FIFO server cannot borrow more than this rate. Besides that, the FIFO server does not have to use up all the borrowing rate

5.1.2.3 Link Bandwidth

In this simulation testing, there are three link bandwidth can be chosen ranging from 1.5 Mbps, 3.0 Mbps and 10 Mbps. These three link bandwidth were chosen because they are usually used at the experimental testing for the traffic scheduling algorithm [6]. If a link bandwidth of 1.5 Mbps is tested, the link can support up to 1500000 bits in one second. Thus, the sources have to send 3000 packets a second to the destination to cause the link situation becomes overloaded or congested (take note here, one packet is 1000 bits).

Similarly, in the normal utilization test, the sources have to generate 1500 packets a second and in the lightly utilization test, the sources have to generate 750 packets a second

For the 3.0 Mbps and 10 Mbps links, the algorithm or calculation is similar to the 1.5 Mbps link. The overall tests that are conducted in this simulation are shown below :

Mechanisms	Bandwidth allocation for CL service source (in %)	Bandwidth allocation for sources BE service source (in %)	Borrowing rate	Link Bandwidth	Link Utilization
Borrow	80	20	10% 20% 30%	1.5 Mbps 3.0 Mbps 10.0 Mbps	Over Normal Light
	70	30			
	60	40			
	50	50			
Without Borrow	80	20	Without borrow rate	1.5 Mbps 3.0 Mbps 10.0 Mbps	Over Normal Light
	70	30			
	60	40			
	50	50			

where CL denotes Controlled Load and BE denotes Best Effort. We use CL and BE in the future to denote the Controlled Load service and Best Effort service.

Table 5.2 Overall Experiments In This Simulation

5.1.3 Testing Output Parameters

Before we proceed to the results of the tests, there are few output parameters which are necessary to be produced for each test. The parameters are :

a.. Numbers of packets generated by the sources,

b. Numbers of packets dropped,

c. Total bandwidth left at the link,

d. Numbers of packets arrive at the destination

These value are accumulated and printed at the end of the simulations. The maximum duration time for the each test is around 300 seconds.

5.2 Testing Results

5.2.1 Introduction

All tests were divided into two parts: one part with the borrow mechanism and another part without the borrow mechanism. Our main focus is to observe the performance of the link-sharing and traffic scheduling in providing bandwidth guarantee. We also want to observe the performance of the borrow mechanism. The test will compose the usage of link bandwidth of 1.5 Mbps, 3.0 Mbps and 10 Mbps. Different utilization of the link between the two sources were also considered (see figure 5.2 for the possibilities). The borrowing rate of 10% , 20% and 30% also will be tested. We group the tests under the different link bandwidth i.e 1.5 Mbps and 10 Mbps. For instance, in the 1.5 Mbps, there are different bandwidth allocation such as 80% to CL, 20% to BE or 70% to CL, 30% to BE, etc. Inside the bandwidth allocation, there are also have borrow mechanism activated and without borrow mechanism. See figure below for further details :

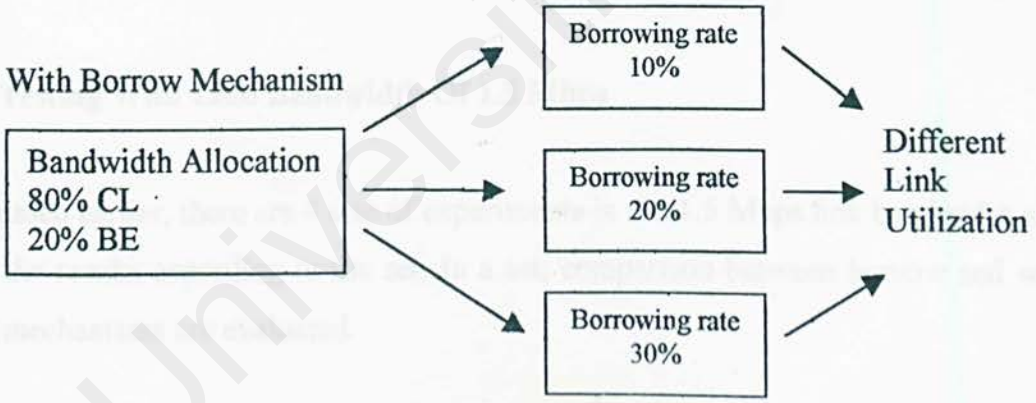


Figure 5.3(cont..) The grouping of test under the link bandwidth of 1.5Mbps

Without Borrow Mechanism

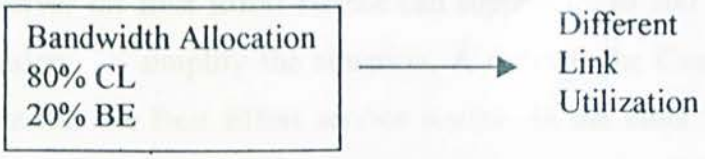


Figure 5.3(cont..) The grouping of test under the link bandwidth of 1.5Mbps

We group above example as one set in the link bandwidth of 1.5 Mbps according to the bandwidth allocation of 80% to CL and 20% for BE. Hence in the link bandwidth of 1.5 Mbps we have 4 sets of different bandwidth allocation. There are listed below :

1) 80% to CL and 20% to BE
2) 70% to CL and 30% to BE
3) 60% to CL and 40% to BE
4) 50% to CL and 50% to BE

All the sets above are tested under different kinds of link utilization.

5.2.1.1 Testing With Link Bandwidth Of 1.5 Mbps

As discussed earlier, there are 4 sets of experiments in the 1.5 Mbps link bandwidth and we discuss the results according to the set. In a set, comparison between borrow and without borrow mechanisms are evaluated.

5.2.1.1.1 Testing With Bandwidth Allocation Of 80% To CL And 20% To BE

We start with the testing of Controlled Load service utilizing 80% of the overall link bandwidth and the Best Effort service utilizing the rest of the link bandwidth that is 20%. The borrowing rate for Controlled Load service is set at 10% in the first test, 20% in the next test and 30% in the final test when the borrow mechanism is activated. We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism.

The FIFO server for Controlled Load service can handle 1200 packets in one second while the FIFO server for Best Effort service can support up to 300 packets under the 80% and 20% allocation. To simplify the situation, A denotes the Controlled load service source while B denotes the Best Effort service source. In the table 5.3, Over denotes overload situation or congestion, normal means heavy utilize situation and light means normal utilization of bandwidth.

Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.4).

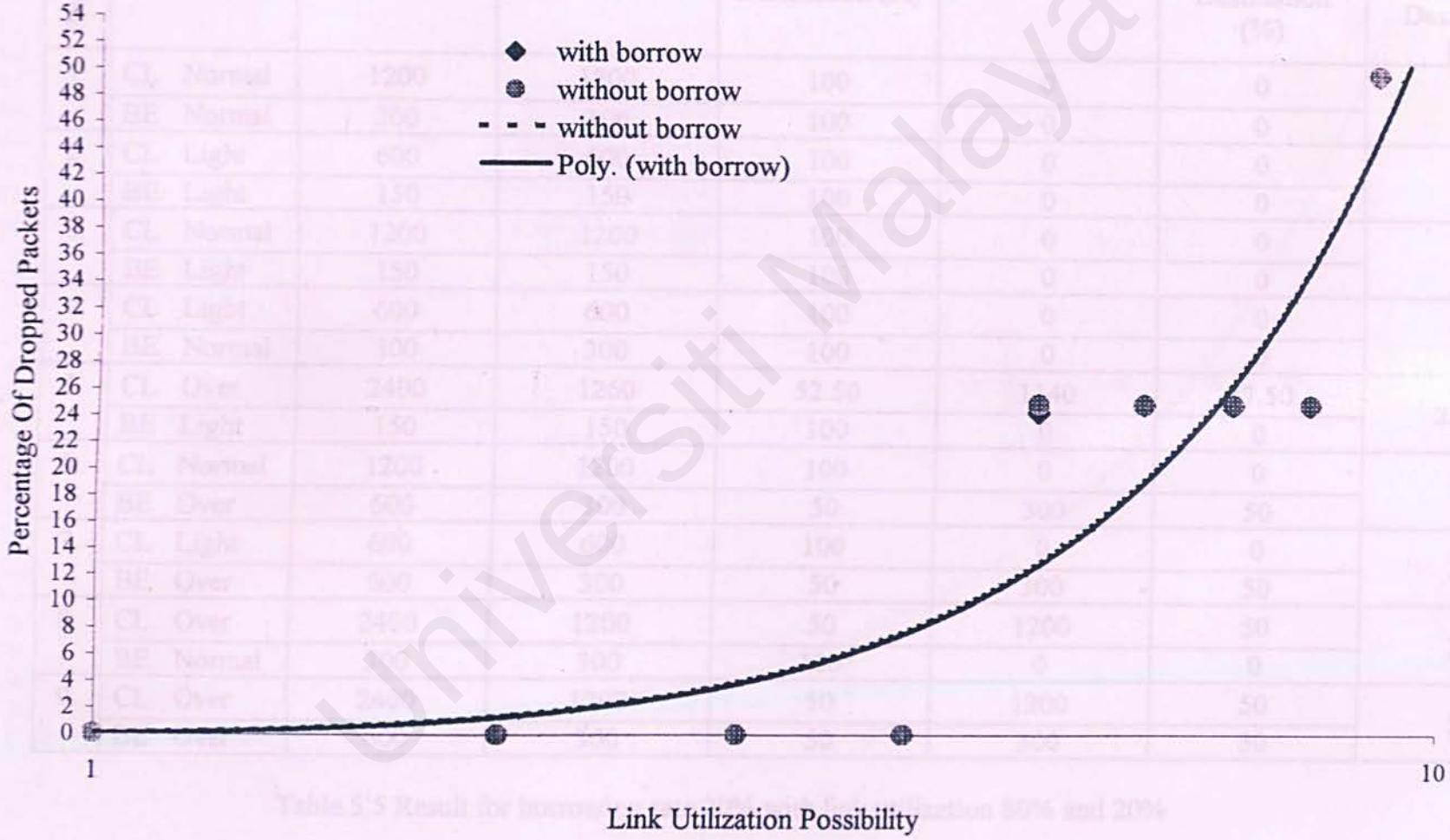
Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1200	1200	100	0	0	0
	BE Normal	300	300	100	0	0	
2.	CL Light	600	600	100	0	0	0
	BE Light	150	150	100	0	0	
3.	CL Normal	1200	1200	100	0	0	0
	BE Light	150	150	100	0	0	
4.	CL Light	600	600	100	0	0	0
	BE Normal	300	300	100	0	0	
5.	CL Over	2400	1230	51.25	1170	48.75	24.38
	BE Light	150	150	100	0	0	
6.	CL Normal	1200	1200	100	0	0	25
	BE Over	600	300	50	300	50	
7.	CL Light	600	600	100	0	0	25
	BE Over	600	300	50	300	50	
8.	CL Over	2400	1200	50	1200	50	25
	BE Normal	300	300	100	0	0	
9.	CL Over	2400	1200	50	1200	50	50
	BE Over	600	300	50	300	50	

Table 5.3 Result for borrowing rate 10% with link utilization 80% for CL and 20% for BE

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1200	1200	100	0	0	0
	BE Normal	300	300	100	0	0	
2.	CL Light	600	600	100	0	0	0
	BE Light	150	150	100	0	0	
3.	CL Normal	1200	1200	100	0	0	0
	BE Light	150	150	100	0	0	
4.	CL Light	600	600	100	0	0	0
	BE Normal	300	300	100	0	0	
5.	CL Over	2400	1200	50	12000	50	25
	BE Light	150	150	100	0	0	
6.	CL Normal	1200	1200	100	0	0	25
	BE Over	600	300	50	300	50	
7.	CL Light	600	600	100	0	0	25
	BE Over	600	300	50	300	50	
8.	CL Over	2400	1200	50	1200	50	25
	BE Normal	300	300	100	0	0	
9.	CL Over	2400	1200	50	1200	50	50
	BE Over	600	300	50	300	50	

Table 5.4 Result for link utilization 80% to CL and 20% to BE without borrow mechanism

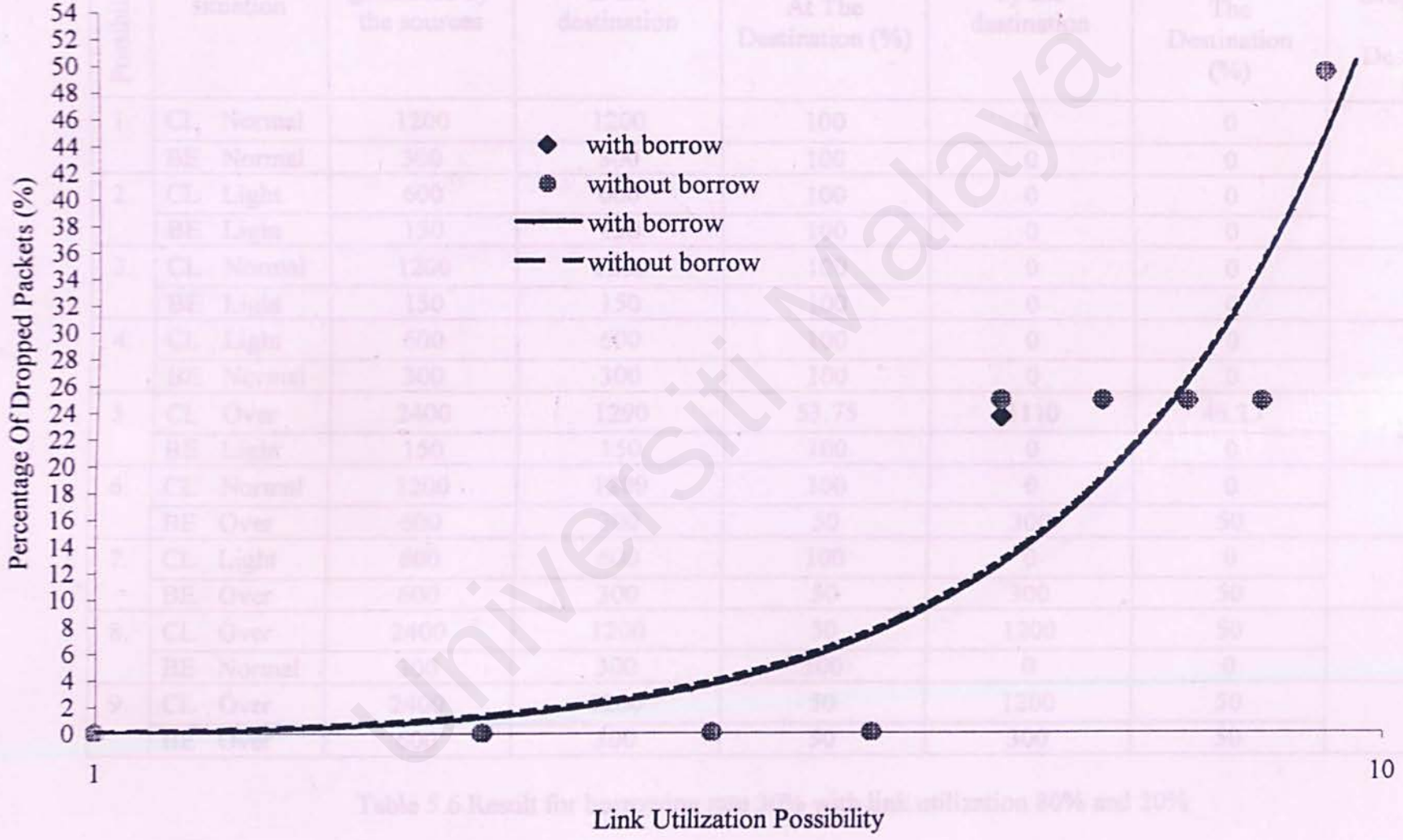
Figure 5.4 Comparison Between The Borrow And Without The Borrow Mechanism At 10% Borrow Rate (Link Utilization Of 80% CL And 20% Of BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1200	1200	100	0	0	0
	BE Normal	300	300	100	0	0	
2.	CL Light	600	600	100	0	0	0
	BE Light	150	150	100	0	0	
3.	CL Normal	1200	1200	100	0	0	0
	BE Light	150	150	100	0	0	
4.	CL Light	600	600	100	0	0	0
	BE Normal	300	300	100	0	0	
5.	CL Over	2400	1260	52.50	1140	47.50	23.75
	BE Light	150	150	100	0	0	
6.	CL Normal	1200	1200	100	0	0	25
	BE Over	600	300	50	300	50	
7.	CL Light	600	600	100	0	0	25
	BE Over	600	300	50	300	50	
8.	CL Over	2400	1200	50	1200	50	25
	BE Normal	300	300	100	0	0	
9.	CL Over	2400	1200	50	1200	50	50
	BE Over	600	300	50	300	50	

Table 5.5 Result for borrowing rate 20% with link utilization 80% and 20%

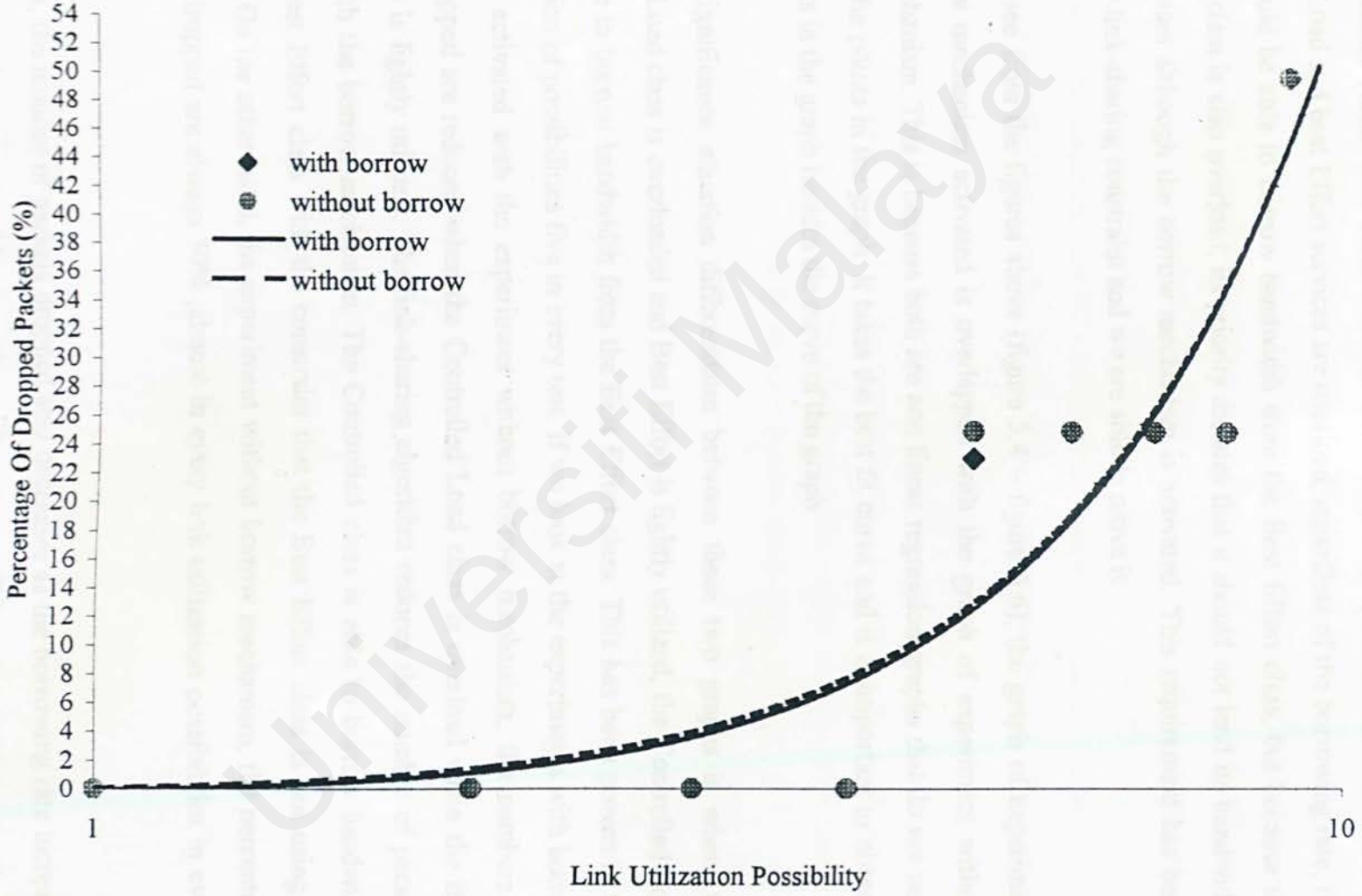
Figure 5.5 Comparison Between The Borrow And Without The Borrow Mechanism At 20% Borrowing Rate (Link Utilization Of 80% CL And 20% Of BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1200	1200	100	0	0	0
	BE Normal	300	300	100	0	0	
2.	CL Light	600	600	100	0	0	0
	BE Light	150	150	100	0	0	
3.	CL Normal	1200	1200	100	0	0	0
	BE Light	150	150	100	0	0	
4.	CL Light	600	600	100	0	0	0
	BE Normal	300	300	100	0	0	
5.	CL Over	2400	1290	53.75	1110	46.25	23.13
	BE Light	150	150	100	0	0	
6.	CL Normal	1200	1200	100	0	0	25
	BE Over	600	300	50	300	50	
7.	CL Light	600	600	100	0	0	25
	BE Over	600	300	50	300	50	
8.	CL Over	2400	1200	50	1200	50	25
	BE Normal	300	300	100	0	0	
9.	CL Over	2400	1200	50	1200	50	50
	BE Over	600	300	50	300	50	

Table 5.6 Result for borrowing rate 30% with link utilization 80% and 20%

Figure 5.6 Comparison Between The Borrow And The Without Borrow Mechanism At 30% Borrow Rate (Link Utilization Of 80% CL And 20% Of BE)



5.2.1.1.2 Discussion

From the results (table 5.3, table 5.5 and table 5.6) above, when the utilization by Controlled Load and best Effort services are overload, regardless of the borrowing rate, the packets should be able to borrow bandwidth from the Best Effort class, but because the Best Effort class is also overload, its priority dictates that it should not lend its bandwidth to other classes although the borrow mechanism is activated. This requirement has been stated in the link-sharing constraint and we are able to prove it.

As we can see from the figures above (figure 5.4 – figure 5.6), the graph of experiment with borrow mechanism activated is overlapped with the graph of experiment without borrow mechanism. This is because both are non linear regression graphs that do not need to pass all the points in the graph. It takes the best fit curve and it is important to observe all the points in the graph besides the curve of the graph

The only significance situation differentiates between these two graphs is where the Controlled Load class is overloaded and Best Effort is lightly utilized, the Controlled Load class is able to borrow bandwidth from the Best Effort class. This has been proven in the link utilization of possibilities five in every test. If we look at the experiments with borrow mechanism activated with the experiment without borrow mechanism, the numbers of packets dropped are reduced when the Controlled Load class is overload while the Best Effort class is lightly utilized. The link-sharing algorithm reduces the number of packets dropped with the borrow mechanism. The Controlled class is able to borrow bandwidth from the Best Effort class with the constraint that the Best Effort class is not using its bandwidth. On the other hand, the experiment without borrow mechanism, the percentage of packets dropped are always 50% ,almost in every link utilization possibilities in every test.

Besides that, the number of packets dropped also decreases as the borrowing rate increases although its quite unnoticeable in these few tests. This once again proves the efficiency of the borrow mechanism.

5.2.1.1.3 Testing With Bandwidth Allocation Of 70% To CL And 30% To BE

The next experiment is the Controlled Load service utilizing 70% of the overall link bandwidth and the Best Effort service utilizing the rest of the link bandwidth that is 30%. The borrowing rate for Controlled Load service is set at 10% in the first test, 20% in the second test and 30% in the final test. The FIFO server for Controlled Load service can handle 1050 packets in one second while the FIFO server for Best Effort service can support up to 450 packets under the 70% and 30% allocation. To simplify the situation, A represents the Controlled load service source while B represents the Best Effort service source.

We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism. Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.8).

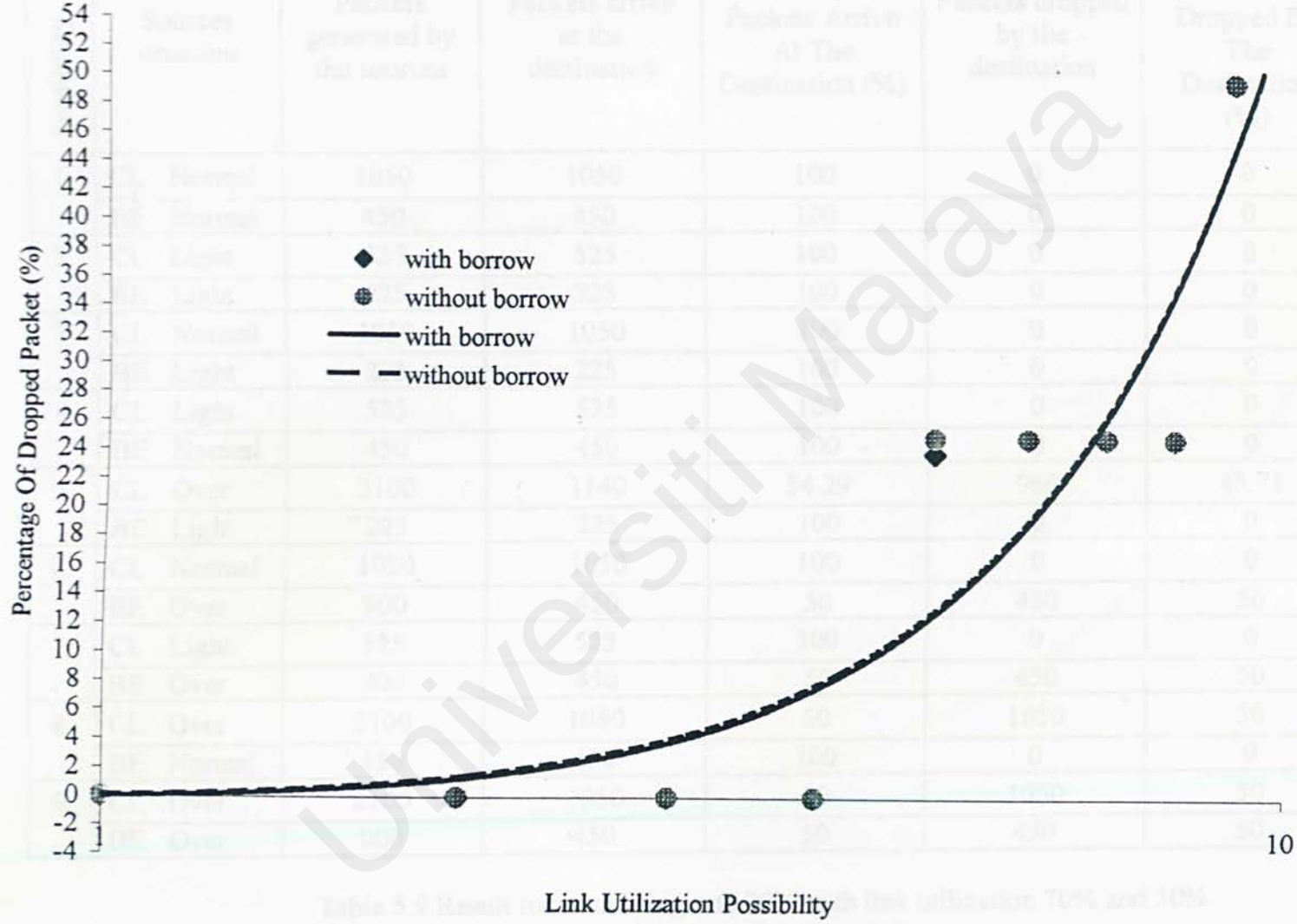
Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1050	1050	100	0	0	0
	BE Normal	450	450	100	0	0	
2.	CL Light	525	525	100	0	0	0
	BE Light	225	225	100	0	0	
3.	CL Normal	1050	1050	100	0	0	0
	BE Light	225	225	100	0	0	
4.	CL Light	525	525	100	0	0	0
	BE Normal	450	450	100	0	0	
5.	CL Over	2100	1095	52.14	1005	46.25	23.93
	BE Light	225	225	100	0	0	
6.	CL Normal	1050	1050	100	0	0	25
	BE Over	900	450	50	450	50	
7.	CL Light	525	525	100	0	0	25
	BE Over	900	450	50	450	50	
8.	CL Over	2100	1050	50	1050	50	25
	BE Normal	450	450	100	0	0	
9.	CL Over	2100	1050	50	1050	50	50
	BE Over	900	450	50	450	50	

Table 5.7 Result for borrowing rate 10% with link utilization 70% and 30%

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1050	1050	100	0	0	0
	BE Normal	450	450	100	0	0	
2.	CL Light	525	525	100	0	0	0
	BE Light	225	225	100	0	0	
3.	CL Normal	1050	1050	100	0	0	0
	BE Light	225	225	100	0	0	
4.	CL Light	525	525	100	0	0	0
	BE Normal	450	450	100	0	0	
5.	CL Over	2100	1050	50	1050	50	25.00
	BE Light	225	225	100	0	0	
6.	CL Normal	1050	1050	100	0	0	25
	BE Over	900	450	50	450	50	
7.	CL Light	525	525	100	0	0	25
	BE Over	900	450	50	450	50	
8.	CL Over	2100	1050	50	1050	50	25
	BE Normal	450	450	100	0	0	
9.	CL Over	2100	1050	50	1050	50	50
	BE Over	900	450	50	450	50	

Table 5.8 Result for link utilization 70% to CL and 30% to BE without borrow mechanism

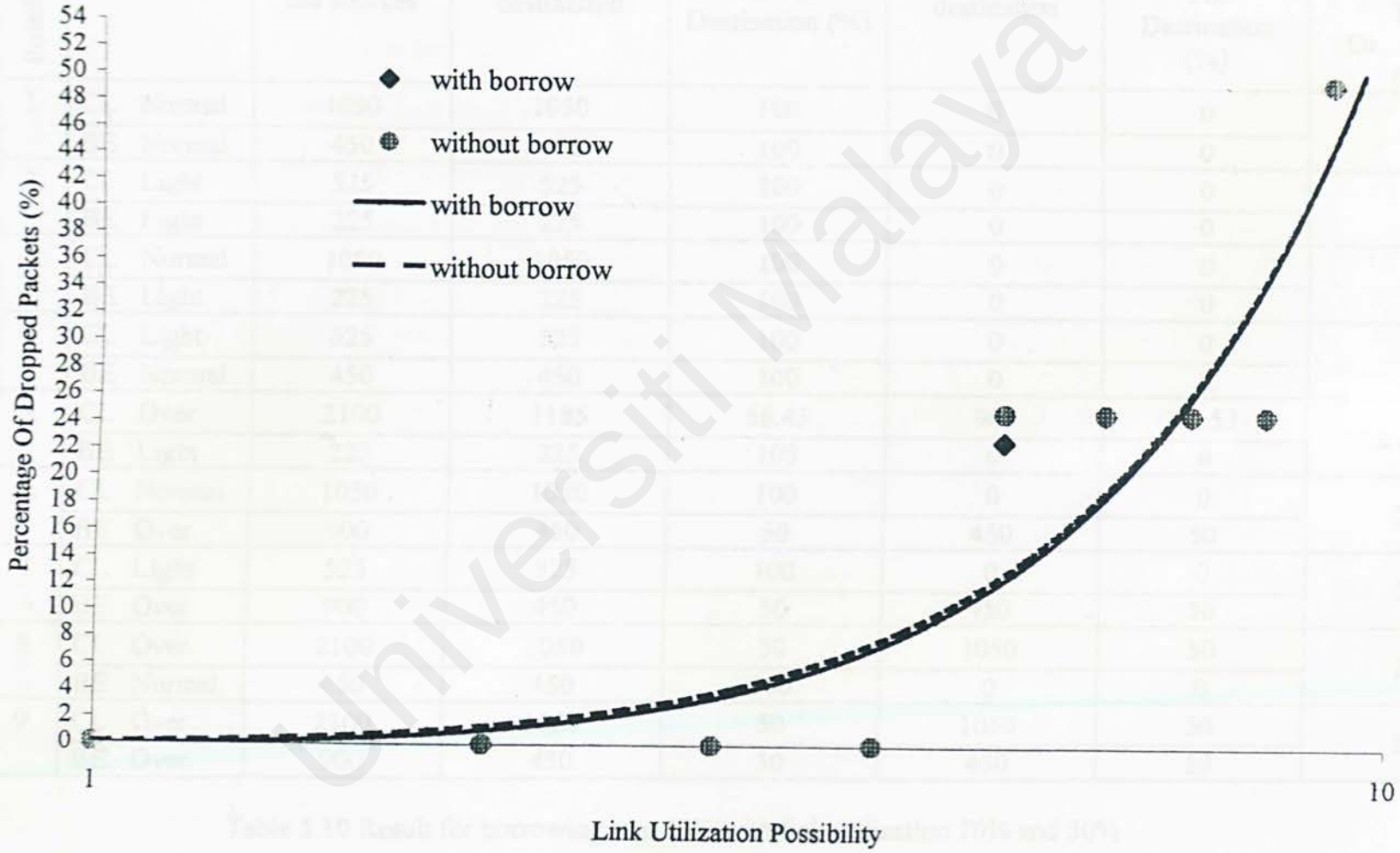
Figure 5.7 Comparison Between The Borrow Mechanism And Without The Borrow Mechanism At 10% Borrow Rate (Link Utilization of 70% CL and 30% of BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1050	1050	100	0	0	0
	BE Normal	450	450	100	0	0	
2.	CL Light	525	525	100	0	0	0
	BE Light	225	225	100	0	0	
3.	CL Normal	1050	1050	100	0	0	0
	BE Light	225	225	100	0	0	
4.	CL Light	525	525	100	0	0	0
	BE Normal	450	450	100	0	0	
5.	CL Over	2100	1140	54.29	960	45.71	22.86
	BE Light	225	225	100	0	0	
6.	CL Normal	1050	1050	100	0	0	25
	BE Over	900	450	50	450	50	
7.	CL Light	525	525	100	0	0	25
	BE Over	900	450	50	450	50	
8.	CL Over	2100	1050	50	1050	50	25
	BE Normal	450	450	100	0	0	
9.	CL Over	2100	1050	50	1050	50	50
	BE Over	900	450	50	450	50	

Table 5.9 Result for borrowing rate 20% with link utilization 70% and 30%

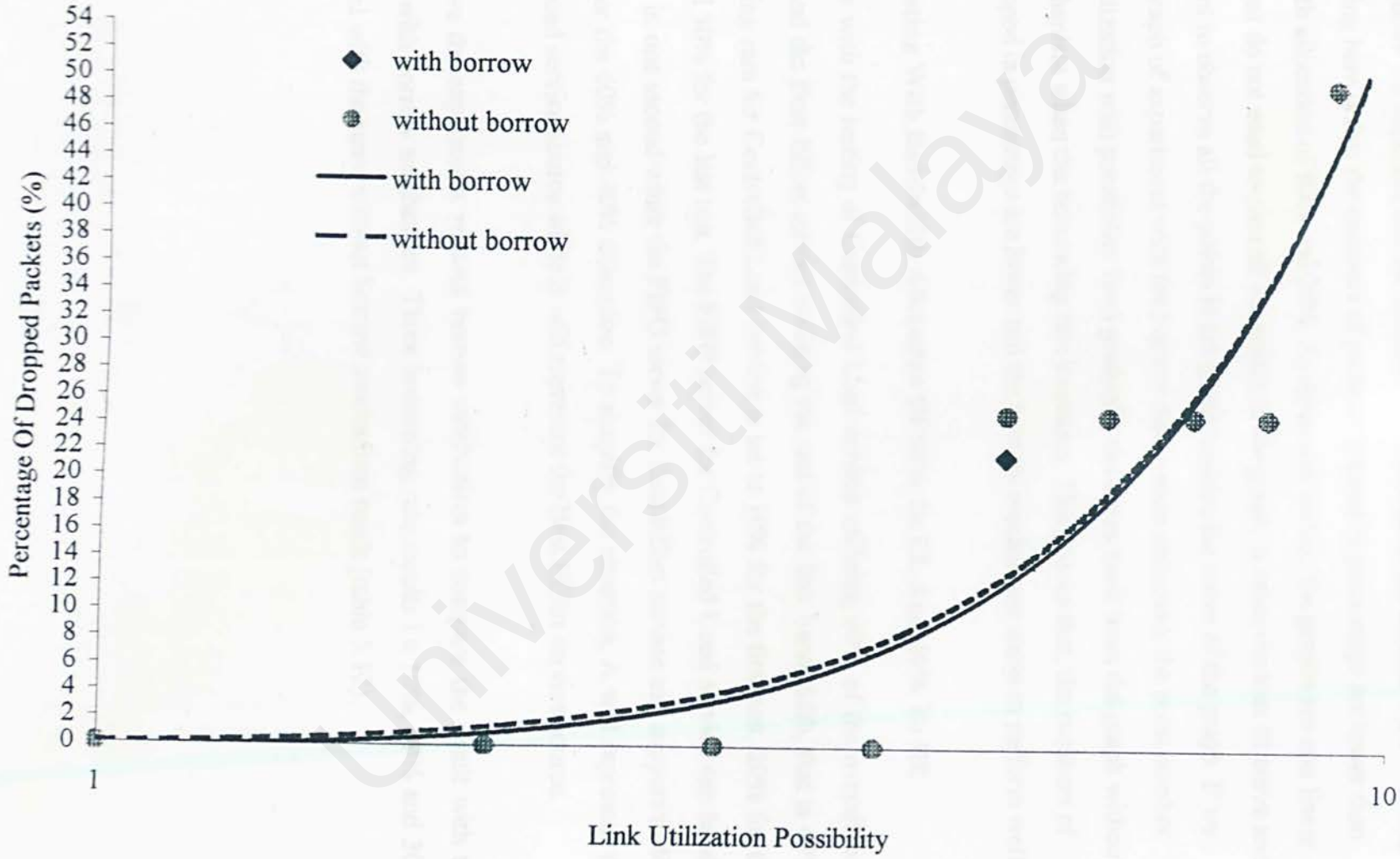
Figure 5.8 Comparison Between The Borrow And Without The Borrow Mechanism At 20% Borrow Rate (Link Utilization Of 70% CL And 30% Of BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	1050	1050	100	0	0	0
	BE Normal	450	450	100	0	0	
2.	CL Light	525	525	100	0	0	0
	BE Light	225	225	100	0	0	
3.	CL Normal	1050	1050	100	0	0	0
	BE Light	225	225	100	0	0	
4.	CL Light	525	525	100	0	0	0
	BE Normal	450	450	100	0	0	
5.	CL Over	2100	1185	56.43	960	43.57	21.79
	BE Light	225	225	100	0	0	
6.	CL Normal	1050	1050	100	0	0	25
	BE Over	900	450	50	450	50	
7.	CL Light	525	525	100	0	0	25
	BE Over	900	450	50	450	50	
8.	CL Over	2100	1050	50	1050	50	25
	BE Normal	450	450	100	0	0	
9.	CL Over	2100	1050	50	1050	50	50
	BE Over	900	450	50	450	50	

Table 5.10 Result for borrowing rate 30% with link utilization 70% and 30%

Figure 5.9 Comparison Between The Borrow And Without The Borrow Mechanism At 30% Borrow Rate (Link Utilization Of 70% CL And 30% Of BE)



5.2.1.1.4 Discussion

The results above are expected and the discussion are similar to the section 5.2.1.1.4. The important thing here is that the numbers of packets dropped in percentage are lower than the bandwidth allocation of 80% and 20%. As discussed earlier, the graphs are non linear regression that do not need to pass all the points in the graph. It takes the best fit curve and it is important to observe all the points in the graph besides the curve of the graph. If we look at the graph of experiment with the borrow mechanism activated, the point number five (link utilization with possibility five) gradually distances itself from the graph without borrow mechanism when the borrowing rate increases. This proves that, the numbers of packets dropped in percentage are lower and the borrow mechanism starts to perform well.

5.2.1.1.5 Testing With Bandwidth Allocation Of 60% To CL And 40% To BE

We continue with the testing of Controlled Load service utilizing 60% of the overall link bandwidth and the Best Effort service utilizing the rest of the link bandwidth, that is 40%. The borrowing rate for Controlled Load service is set at 10% for the first test, 20% for the next test and 30% for the last test. The FIFO server for Controlled Load service can handle 900 packets in one second while the FIFO server for Best Effort service can support to 600 packets under the 60% and 40% allocation. To simplify the situation, A will represent the Controlled load service source while B will represent the Best Effort service source.

We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism. Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.12).

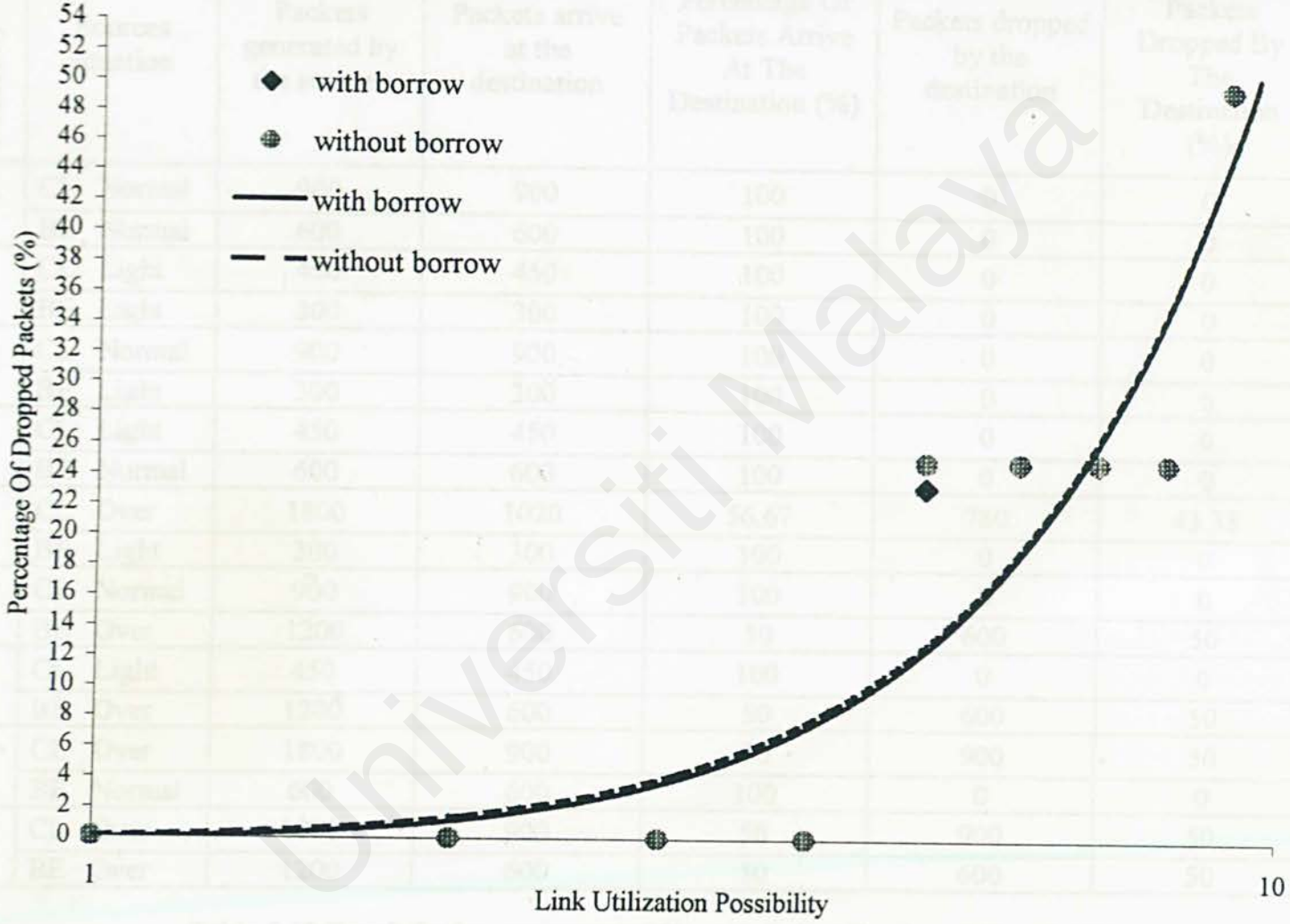
Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	900	900	100	0	0	0
	BE Normal	600	600	100	0	0	
2.	CL Light	450	450	100	0	0	0
	BE Light	300	300	100	0	0	
3.	CL Normal	900	900	100	0	0	0
	BE Light	300	300	100	0	0	
4.	CL Light	450	450	100	0	0	0
	BE Normal	600	600	100	0	0	
5.	CL Over	1800	960	53.33	840	46.67	23.34
	BE Light	300	300	100	0	0	
6.	CL Normal	900	900	100	0	0	25
	BE Over	1200	600	50	600	50	
7.	CL Light	450	450	100	0	0	25
	BE Over	1200	600	50	600	50	
8.	CL Over	1800	900	50	900	50	25
	BE Normal	600	600	100	0	0	
9.	CL Over	1800	900	50	900	50	50
	BE Over	1200	600	50	600	50	

Table 5.11 Result for borrowing rate 10% with link utilization 60% and 40%

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	900	900	100	0	0	0
	BE Normal	600	600	100	0	0	
2.	CL Light	450	450	100	0	0	0
	BE Light	300	300	100	0	0	
3.	CL Normal	900	900	100	0	0	0
	BE Light	300	300	100	0	0	
4.	CL Light	450	450	100	0	0	0
	BE Normal	600	600	100	0	0	
5.	CL Over	1800	900	50	900	50	25
	BE Light	300	300	100	0	0	
6.	CL Normal	900	900	100	0	0	25
	BE Over	1200	600	50	600	50	
7.	CL Light	450	450	100	0	0	25
	BE Over	1200	600	50	600	50	
8.	CL Over	1800	900	50	900	50	25
	BE Normal	600	600	100	0	0	
9.	CL Over	1800	900	50	900	50	50
	BE Over	1200	600	50	600	50	

Table 5.12 Result for link utilization 60% to CL and 40% to BE without borrow mechanism

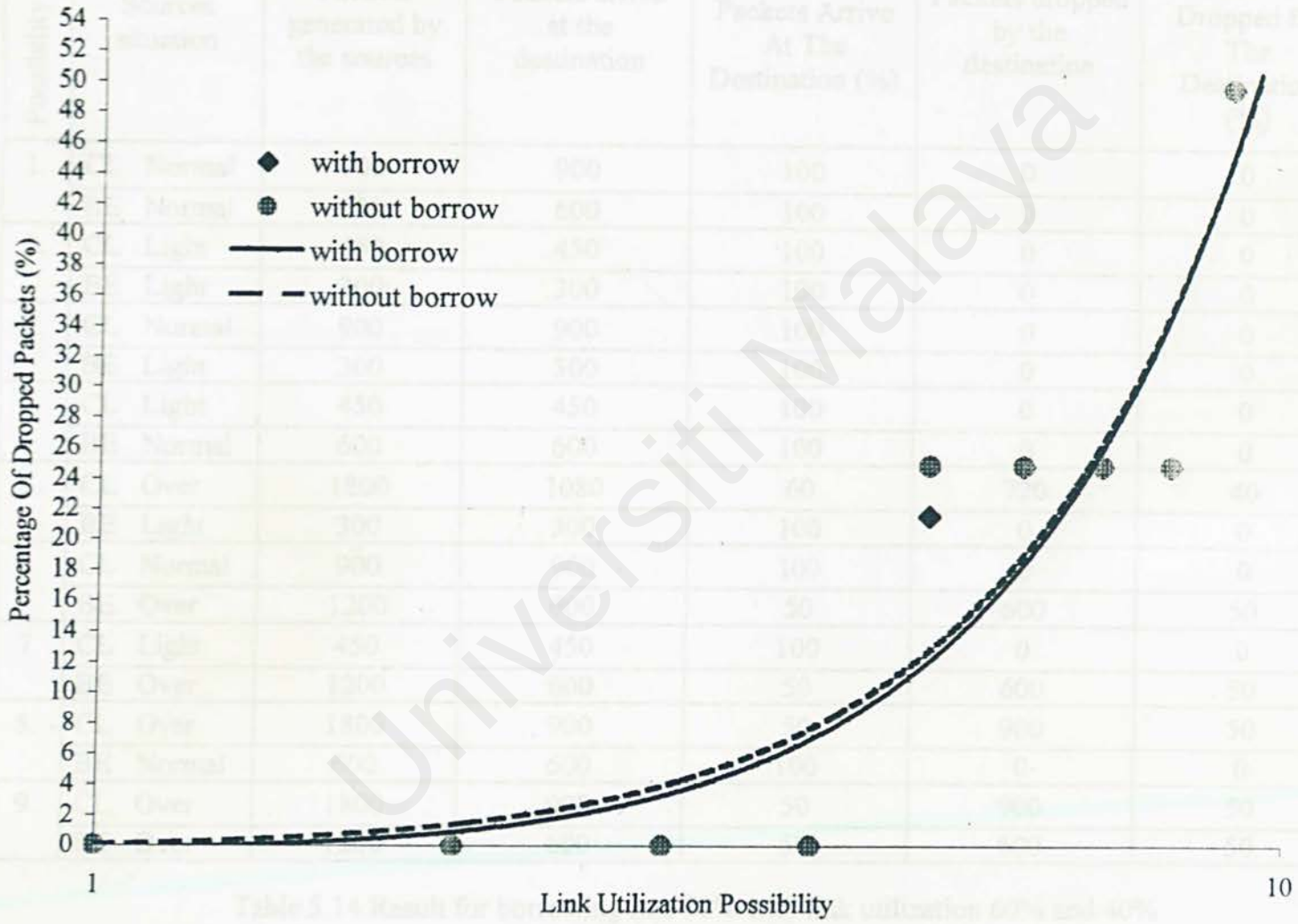
Figure 5.10 Comparison Between The Borrow And Without The Borrow Mechanism At 10% Borrow Rate
(Link Utilization Of 60% CL And 40% Of BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	900	900	100	0	0	0
	BE Normal	600	600	100	0	0	
2.	CL Light	450	450	100	0	0	0
	BE Light	300	300	100	0	0	
3.	CL Normal	900	900	100	0	0	0
	BE Light	300	300	100	0	0	
4.	CL Light	450	450	100	0	0	0
	BE Normal	600	600	100	0	0	
5.	CL Over	1800	1020	56.67	780	43.33	21.67
	BE Light	300	300	100	0	0	
6.	CL Normal	900	900	100	0	0	25
	BE Over	1200	600	50	600	50	
7.	CL Light	450	450	100	0	0	25
	BE Over	1200	600	50	600	50	
8.	CL Over	1800	900	50	900	50	25
	BE Normal	600	600	100	0	0	
9.	CL Over	1800	900	50	900	50	50
	BE Over	1200	600	50	600	50	

Table 5.13 Result for borrowing rate 20% with link utilization 60% and 40%

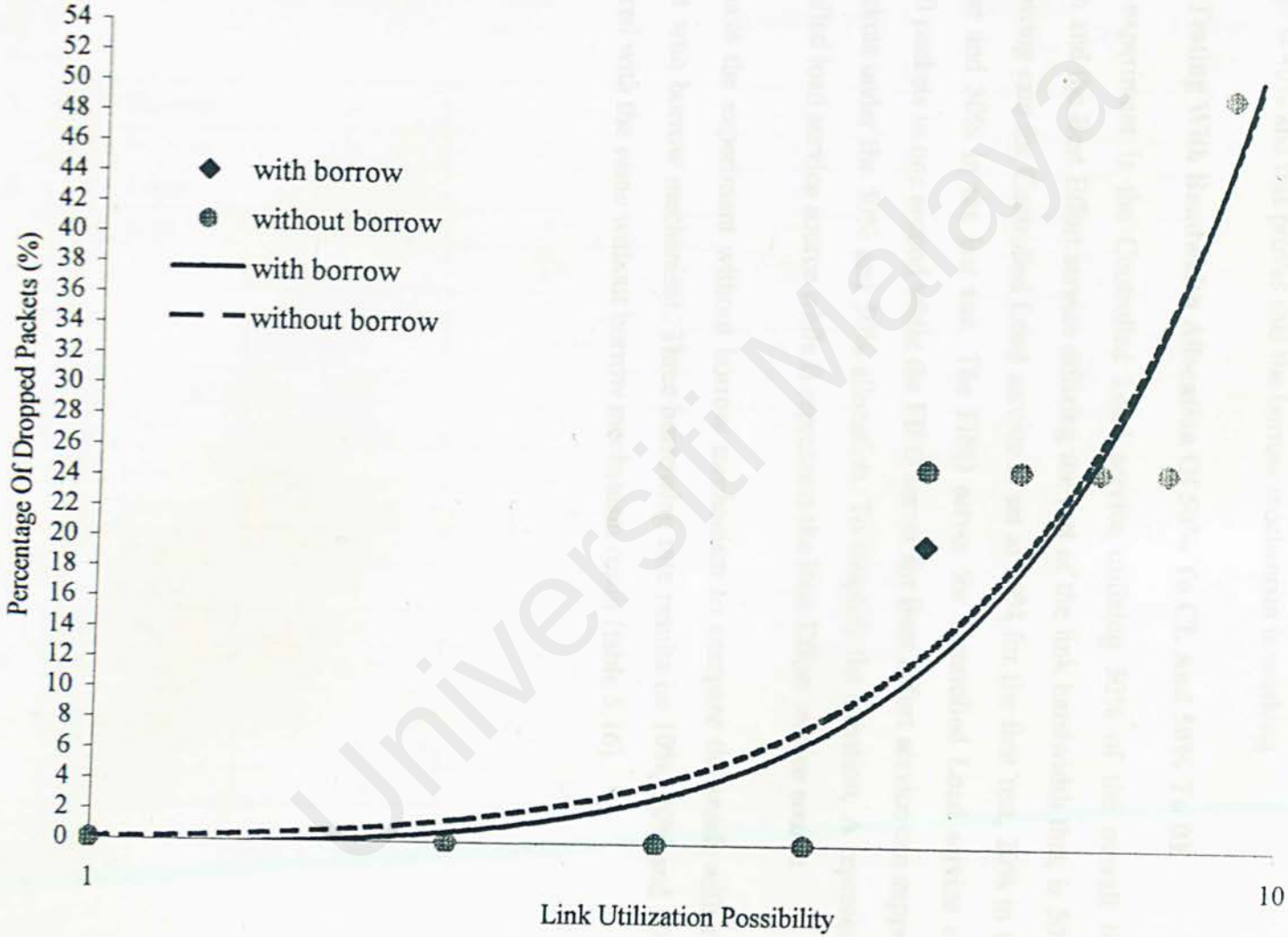
Figure 5.11 Comparison Between The Borrow And Without Borrow Mechanism At 20% Borrow Rate (Link Utilization Of 60% CL And 40% Of BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	900	900	100	0	0	0
	BE Normal	600	600	100	0	0	
2.	CL Light	450	450	100	0	0	0
	BE Light	300	300	100	0	0	
3.	CL Normal	900	900	100	0	0	0
	BE Light	300	300	100	0	0	
4.	CL Light	450	450	100	0	0	0
	BE Normal	600	600	100	0	0	
5.	CL Over	1800	1080	60	720	40	20
	BE Light	300	300	100	0	0	
6.	CL Normal	900	900	100	0	0	25
	BE Over	1200	600	50	600	50	
7.	CL Light	450	450	100	0	0	25
	BE Over	1200	600	50	600	50	
8.	CL Over	1800	900	50	900	50	25
	BE Normal	600	600	100	0	0	
9.	CL Over	1800	900	50	900	50	50
	BE Over	1200	600	50	600	50	

Table 5.14 Result for borrowing rate 30% with link utilization 60% and 40%

Figure 5.14 Comparison Between The Borrow And Without The Borrow Mechanism At 30% borrow Rate (Link Utilization Of 60% CL And 40% Of BE)



5.2.1.1.6 Discussion

The results above are expected and the discussion are similar to the section 5.2.1.1.4. The important thing here is that the numbers of packets dropped in percentage are lower than the bandwidth allocation of 70% and 30%. The lowest numbers of packet dropped in percentage is 40% and this proves that the borrow mechanism is working

5.2.1.1.7 Testing With Bandwidth Allocation Of 50% To CL And 50% To BE

The last experiment is the Controlled Load service utilizing 50% of the overall link bandwidth and the Best Effort service utilizing the rest of the link bandwidth that, is 50%. The borrowing rate for Controlled Load service is set at 10% for the first test, 20% in the second test and 30% in the last test. The FIFO server for Controlled Load service can handle 750 packets in one second while the FIFO server for Best Effort service can support to 750 packets under the 50% and 50% allocation. To simplify the situation, A represents the Controlled load service source while B represents the Best Effort service source.

We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism. Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.16).

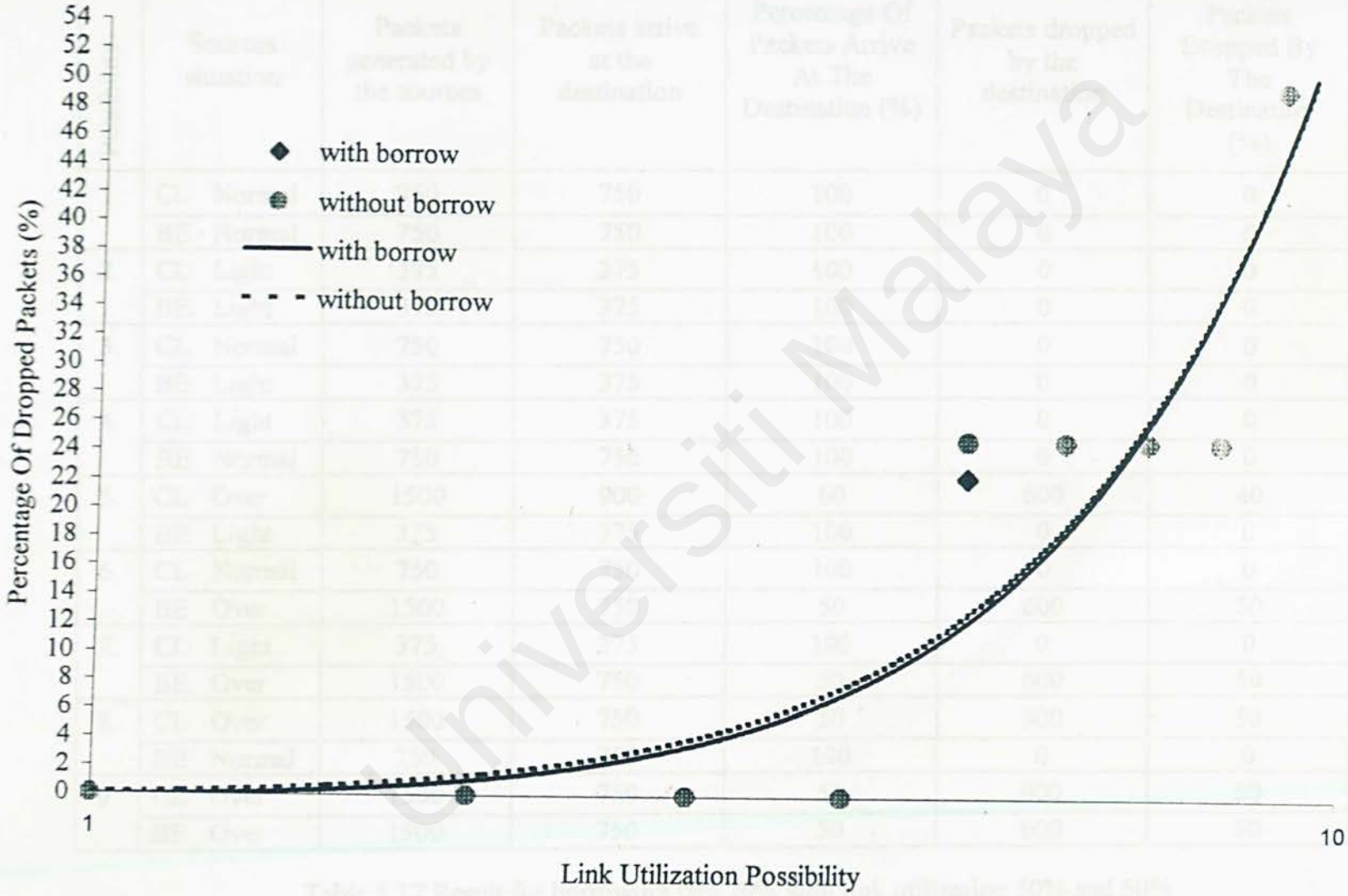
Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	750	750	100	0	0	0
	BE Normal	750	750	100	0	0	
2.	CL Light	375	375	100	0	0	0
	BE Light	375	375	100	0	0	
3.	CL Normal	750	750	100	0	0	0
	BE Light	375	375	100	0	0	
4.	CL Light	375	375	100	0	0	0
	BE Normal	750	750	100	0	0	
5.	CL Over	1500	825	55	675	45	22.50
	BE Light	375	375	100	0	0	
6.	CL Normal	750	750	100	0	0	25
	BE Over	1500	750	50	600	50	
7.	CL Light	375	375	100	0	0	25
	BE Over	1500	750	50	600	50	
8.	CL Over	1500	750	50	900	50	25
	BE Normal	750	750	100	0	0	
9.	CL Over	1500	750	50	900	50	50
	BE Over	1500	750	50	600	50	

Table 5.15 Result for borrowing rate 10% with link utilization 50% and 50%

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	750	750	100	0	0	0
	BE Normal	750	750	100	0	0	
2.	CL Light	375	375	100	0	0	0
	BE Light	375	375	100	0	0	
3.	CL Normal	750	750	100	0	0	0
	BE Light	375	375	100	0	0	
4.	CL Light	375	375	100	0	0	0
	BE Normal	750	750	100	0	0	
5.	CL Over	1500	750	50	750	50	25
	BE Light	375	375	100	0	0	
6.	CL Normal	750	750	100	0	0	25
	BE Over	1500	750	50	600	50	
7.	CL Light	375	375	100	0	0	25
	BE Over	1500	750	50	600	50	
8.	CL Over	1500	750	50	900	50	25
	BE Normal	750	750	100	0	0	
9.	CL Over	1500	750	50	900	50	50
	BE Over	1500	750	50	600	50	

Table 5.16 Result for link utilization 50% to CL and 50% to BE without borrow mechanism

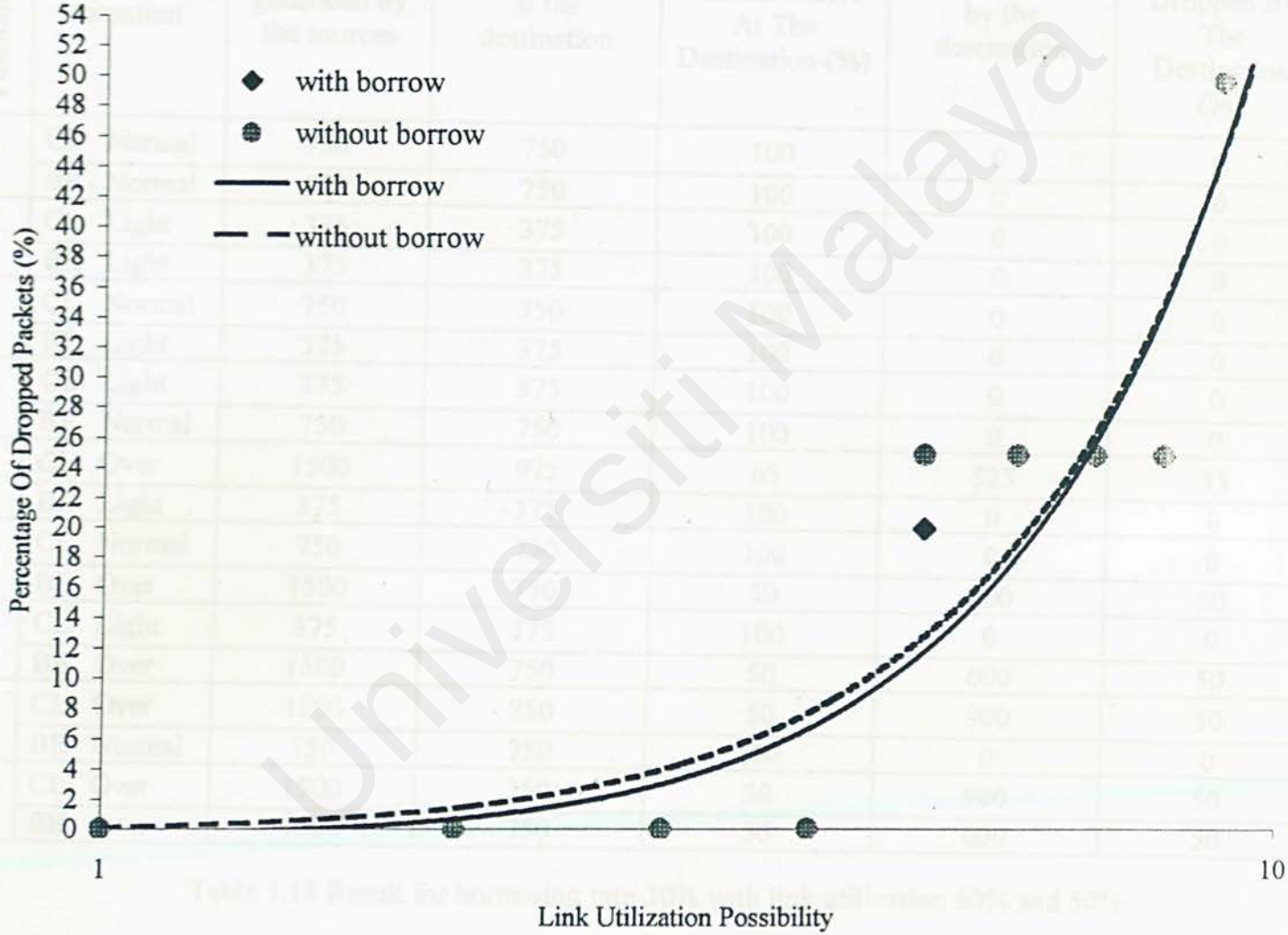
Figure 5.13 Comparison Between The Borrow And Without The Borrow Mechanism At 10% Borrow Rate (Link Utilization Of 50% CL And 50% Of BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	750	750	100	0	0	0
	BE Normal	750	750	100	0	0	
2.	CL Light	375	375	100	0	0	0
	BE Light	375	375	100	0	0	
3.	CL Normal	750	750	100	0	0	0
	BE Light	375	375	100	0	0	
4.	CL Light	375	375	100	0	0	0
	BE Normal	750	750	100	0	0	
5.	CL Over	1500	900	60	600	40	20
	BE Light	375	375	100	0	0	
6.	CL Normal	750	750	100	0	0	25
	BE Over	1500	750	50	600	50	
7.	CL Light	375	375	100	0	0	25
	BE Over	1500	750	50	600	50	
8.	CL Over	1500	750	50	900	50	25
	BE Normal	750	750	100	0	0	
9.	CL Over	1500	750	50	900	50	50
	BE Over	1500	750	50	600	50	

Table 5.17 Result for borrowing rate 20% with link utilization 50% and 50%

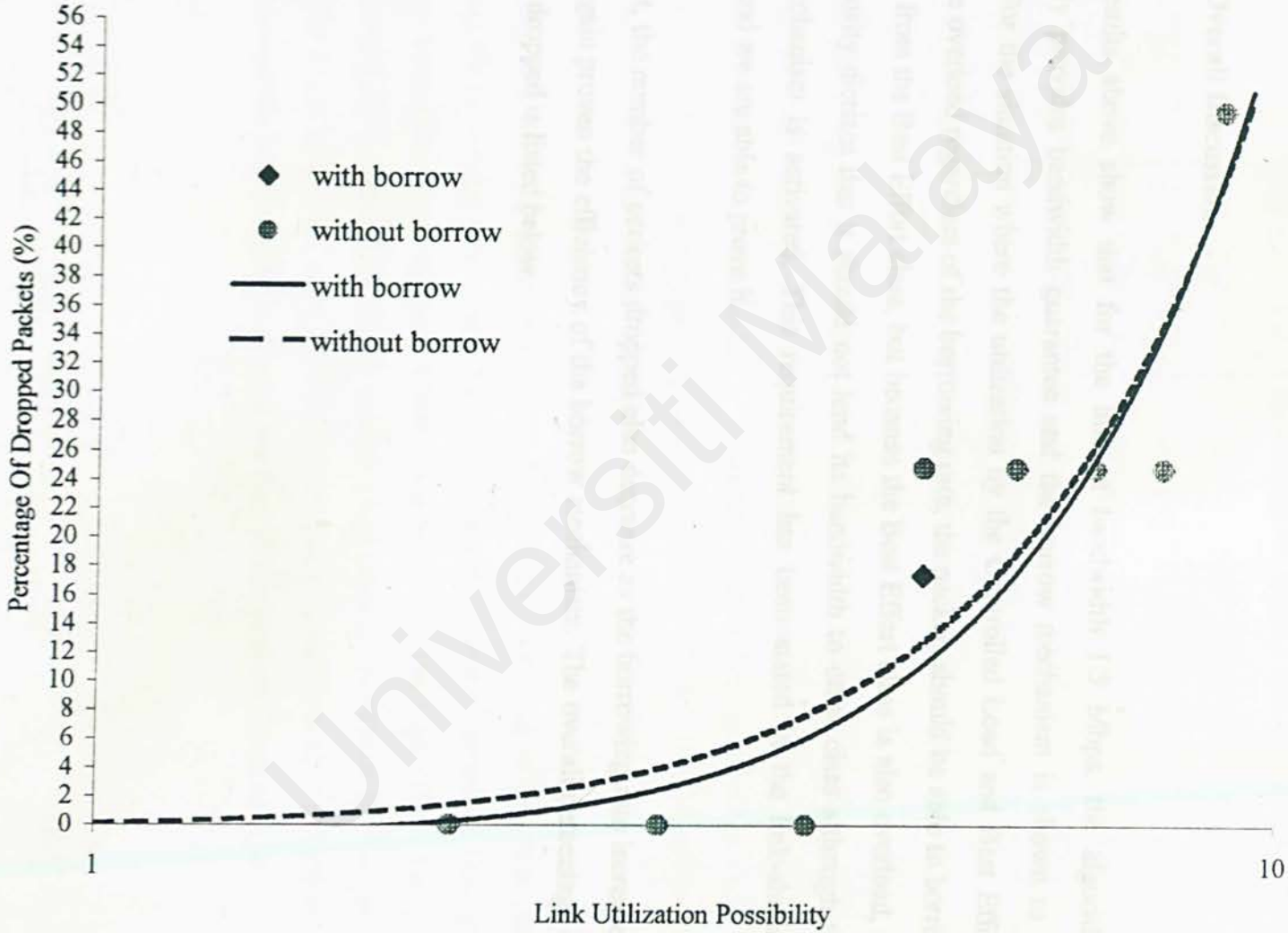
Figure 5.14 Comparison Betweem The Borrow And Without The Borrow Mechanism At 20% Borrow Rate (Link Utilization Of 50% CL And 50% BE)



Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	750	750	100	0	0	0
	BE Normal	750	750	100	0	0	
2.	CL Light	375	375	100	0	0	0
	BE Light	375	375	100	0	0	
3.	CL Normal	750	750	100	0	0	0
	BE Light	375	375	100	0	0	
4.	CL Light	375	375	100	0	0	0
	BE Normal	750	750	100	0	0	
5.	CL Over	1500	975	65	525	35	17.5
	BE Light	375	375	100	0	0	
6.	CL Normal	750	750	100	0	0	25
	BE Over	1500	750	50	600	50	
7.	CL Light	375	375	100	0	0	25
	BE Over	1500	750	50	600	50	
8.	CL Over	1500	750	50	900	50	25
	BE Normal	750	750	100	0	0	
9.	CL Over	1500	750	50	900	50	50
	BE Over	1500	750	50	600	50	

Table 5.18 Result for borrowing rate 30% with link utilization 50% and 50%

Figure 5.15 Comparison Between The Borrow And Without The Borrow Mechanism At 30% Borrow Rate (Link Utilization Of 50% CL And 50% Of BE)



5.2.1.1.8 Discussion

The results above are expected and the discussion are similar to the section 5.2.1.1.4. The important thing here is that the numbers of packets dropped in percentage are the lowest among the three sets of bandwidth allocation.

5.2.1.1.9 Overall Discussion

All the results above show that for the link of bandwidth 1.5 Mbps, the algorithm successfully provides bandwidth guarantee and the borrow mechanism is shown to be efficient. For the situation where the utilization by the Controlled Load and Best Effort services are overload regardless of the borrowing rate, the packets should be able to borrow bandwidth from the Best Effort class, but because the Best Effort class is also overload, its has the priority dictates that it should not lend its bandwidth to other class although the borrow mechanism is activated. This requirement has been stated in the link-sharing constraint and we are able to prove it.

Besides that, the number of packets dropped also decrease as the borrowing rate increases. This once again proven the efficiency of the borrow mechanism. The overall percentage of the packets dropped is listed below.

Allocation Of Bandwidth To Controlled Load Class (in percentage, %)	Allocation Of Bandwidth To Best Effort Class (in percentage, %)	Packets Dropped By the Simulation (in percentage, %) with 10% borrow rate	Packets Dropped By the Simulation (in percentage, %) with 20% borrow rate	Packets Dropped By the Simulation (in percentage, %) with 30% borrow rate
80	20	48.75	47.50	46.25
70	30	47.86	45.71	43.57
60	40	46.67	43.33	40.00
50	50	45.00	40.00	35.00

Table 5.19 : The overall percentage of packets dropped

The highest percentage of packets dropped is 48.75% and is almost equal to 50%. An allocation of 80% and 20% is not recommended because it may cause many packets to be dropped although the borrow mechanism is activated. On the other hand , if the allocation of bandwidth is 50%, the percentage of packets dropped is 35% for 30% borrowing rate and it is very desirable to have such low of percentage of dropped packets.

We have also test the situation where the Controlled Load and Best effort class facing overload and the borrowing rate is high. Firstly, the packets from the Controlled Load class will borrow bandwidth from the Best Effort class but when the Best Effort class also having overload, the packets from the Controlled Load class is preempted and the packets from the Best Effort class start using the bandwidth. The packets from the Controlled load class will resume the borrowing process after the Best Effort packets finish using it.

5.2.1.2 Testing With Link Bandwidth Of 10 Mbps

As explained in section 5.2.1, there are 4 different bandwidth allocations sets tested in this simulation. Please refer to section 5.2.1 for further details.

5.2.1.2.1 Testing With Bandwidth Allocation Of 80% To CL And 20% To BE (10 Mbps)

Like in section 5.2.1.1, we start testing the Controlled Load service utilizing 80% of the overall link bandwidth and the Best Effort service utilizing 20% of the link bandwidth. The borrowing rate for Controlled Load service is set at 10% in the first test, 20% in the second test and 30% in the last test. The FIFO server for Controlled Load service can handle 8000 packets in one second while the FIFO server for Best Effort service can support up to 2000 packets under the 80% and 20% allocation. To simplify the situation, A represents the Controlled load service source while B represents the Best Effort service source. Take note here, the result of 3 Mbps link bandwidth is not listed because it is exactly the same as the result for the link bandwidth 1.5 Mbps.

We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism. Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.21). We discuss the result at the end of the test because the discussion is exactly similar to the discussion in section 5.2.1.1 and varies in the quantity of packets. Therefore, we evaluate the overall result in the end of this 10 Mbps link bandwidth test. There are also no comparison graphs between the borrow mechanism experiment and without the borrow mechanism experiment because of the same reason given above.

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	8000	8000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
2.	CL Light	4000	4000	100	0	0	0
	BE Light	1000	1000	100	0	0	
3.	CL Normal	8000	8000	100	0	0	0
	BE Light	1000	1000	100	0	0	
4.	CL Light	4000	4000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
5.	CL Over	16000	8200	51.25	7800	48.75	24.38
	BE Light	1000	1000	100	0	0	
6.	CL Normal	8000	8000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
7.	CL Light	4000	4000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
8.	CL Over	16000	8000	50	8000	50	25
	BE Normal	2000	2000	100	0	0	
9.	CL Over	16000	8000	50	8000	50	50
	BE Over	4000	2000	50	2000	50	

Table 5.20 Result for borrowing rate 10% with link utilization 80% and 20% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	8000	8000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
2.	CL Light	4000	4000	100	0	0	0
	BE Light	1000	1000	100	0	0	
3.	CL Normal	8000	8000	100	0	0	0
	BE Light	1000	1000	100	0	0	
4.	CL Light	4000	4000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
5.	CL Over	16000	8000	50	8000	50	24.38
	BE Light	1000	1000	100	0	0	
6.	CL Normal	8000	8000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
7.	CL Light	4000	4000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
8.	CL Over	16000	8000	50	8000	50	25
	BE Normal	2000	2000	100	0	0	
9.	CL Over	16000	8000	50	8000	50	50
	BE Over	4000	2000	50	2000	50	

Table 5.21 Result for link utilization 80% to CL and 20% to BE without borrow mechanism (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	8000	8000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
2.	CL Light	4000	4000	100	0	0	0
	BE Light	1000	1000	100	0	0	
3.	CL Normal	8000	8000	100	0	0	0
	BE Light	1000	1000	100	0	0	
4.	CL Light	4000	4000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
5.	CL Over	16000	8400	52.5	7600	47.5	23.75
	BE Light	1000	1000	100	0	0	
6.	CL Normal	8000	8000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
7.	CL Light	4000	4000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
8.	CL Over	16000	8000	50	8000	50	25
	BE Normal	2000	2000	100	0	0	
9.	CL Over	16000	8000	50	8000	50	50
	BE Over	4000	2000	50	2000	50	

Table 5.22 Result for borrowing rate 20% with link utilization 80% and 20% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	8000	8000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
2.	CL Light	4000	4000	100	0	0	0
	BE Light	1000	1000	100	0	0	
3.	CL Normal	8000	8000	100	0	0	0
	BE Light	1000	1000	100	0	0	
4.	CL Light	4000	4000	100	0	0	0
	BE Normal	2000	2000	100	0	0	
5.	CL Over	16000	8600	53.75	7400	46.25	23.13
	BE Light	1000	1000	100	0	0	
6.	CL Normal	8000	8000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
7.	CL Light	4000	4000	100	0	0	25
	BE Over	4000	2000	50	2000	50	
8.	CL Over	16000	8000	50	8000	50	25
	BE Normal	2000	2000	100	0	0	
9.	CL Over	16000	8000	50	8000	50	50
	BE Over	4000	2000	50	2000	50	

Table 5.23 Result for borrowing rate 30% with link utilization 80% and 20% (10 Mbps)

5.2.1.2.2 Testing With Bandwidth Allocation Of 70% To CL And 30% To BE (10 Mbps)

We continue with the testing of Controlled Load service utilizing 70% of the overall link bandwidth and the Best Effort service utilizing the rest of the link bandwidth that is 30%. The borrowing rate for Controlled Load service is set at 10% for the first test, 20% for the next test and 30% for the last test. The FIFO server for Controlled Load service can handle 7000 packets in one second while the FIFO server for Best Effort service can support to 3000 packets under the 70% and 30% allocation. To simplify the situation, A will represent the Controlled load service source while B will represent the Best Effort service source.

We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism. Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.25).

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	7000	7000	100	0	0	0
	BE Normal	3000	3000	100	0	0	
2.	CL Light	3500	3500	100	0	0	0
	BE Light	1500	1500	100	0	0	
3.	CL Normal	7000	7000	100	0	0	0
	BE Light	1500	1500	100	0	0	
4.	CL Light	3500	3500	100	0	0	0
	BE Normal	3000	3000	100	0	0	
5.	CL Over	14000	7300	52.14	6700	47.86	23.93
	BE Light	1500	1500	100	0	0	
6.	CL Normal	7000	7000	100	0	0	25
	BE Over	6000	3000	50	3000	50	
7.	CL Light	3500	3500	100	0	0	25
	BE Over	6000	3000	50	3000	50	
8.	CL Over	14000	7000	50	7000	50	25
	BE Normal	3000	3000	100	0	0	
9.	CL Over	14000	7000	50	7000	50	50
	BE Over	6000	3000	50	3000	50	

Table 5.24 Result for borrowing rate 10% with link utilization 70% and 30% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	7000	7000	100	0	0	0
	BE Normal	3000	3000	100	0	0	
2.	CL Light	3500	3500	100	0	0	0
	BE Light	1500	1500	100	0	0	
3.	CL Normal	7000	7000	100	0	0	0
	BE Light	1500	1500	100	0	0	
4.	CL Light	3500	3500	100	0	0	0
	BE Normal	3000	3000	100	0	0	
5.	CL Over	14000	7000	50	7000	50	25
	BE Light	1500	1500	100	0	0	
6.	CL Normal	7000	7000	100	0	0	25
	BE Over	6000	3000	50	3000	50	
7.	CL Light	3500	3500	100	0	0	25
	BE Over	6000	3000	50	3000	50	
8.	CL Over	14000	7000	50	7000	50	25
	BE Normal	3000	3000	100	0	0	
9.	CL Over	14000	7000	50	7000	50	50
	BE Over	6000	3000	50	3000	50	

Table 5.25 Result for link utilization 70% to CL and 30% to BE without borrow mechanism (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	7000	7000	100	0	0	0
	BE Normal	3000	3000	100	0	0	
2.	CL Light	3500	3500	100	0	0	0
	BE Light	1500	1500	100	0	0	
3.	CL Normal	7000	7000	100	0	0	0
	BE Light	1500	1500	100	0	0	
4.	CL Light	3500	3500	100	0	0	0
	BE Normal	3000	3000	100	0	0	
5.	CL Over	14000	7600	54.29	6400	45.71	22.86
	BE Light	1500	1500	100	0	0	
6.	CL Normal	7000	7000	100	0	0	25
	BE Over	6000	3000	50	3000	50	
7.	CL Light	3500	3500	100	0	0	25
	BE Over	6000	3000	50	3000	50	
8.	CL Over	14000	7000	50	7000	50	25
	BE Normal	3000	3000	100	0	0	
9.	CL Over	14000	7000	50	7000	50	50
	BE Over	6000	3000	50	3000	50	

Table 5.26 Result for borrowing rate 20% with link utilization 70% and 30% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	7000	7000	100	0	0	0
	BE Normal	3000	3000	100	0	0	
2.	CL Light	3500	3500	100	0	0	0
	BE Light	1500	1500	100	0	0	
3.	CL Normal	7000	7000	100	0	0	0
	BE Light	1500	1500	100	0	0	
4.	CL Light	3500	3500	100	0	0	0
	BE Normal	3000	3000	100	0	0	
5.	CL Over	14000	7900	56.43	6100	43.57	21.79
	BE Light	1500	1500	100	0	0	
6.	CL Normal	7000	7000	100	0	0	25
	BE Over	6000	3000	50	3000	50	
7.	CL Light	3500	3500	100	0	0	25
	BE Over	6000	3000	50	3000	50	
8.	CL Over	14000	7000	50	7000	50	25
	BE Normal	3000	3000	100	0	0	
9.	CL Over	14000	7000	50	7000	50	50
	BE Over	6000	3000	50	3000	50	

Table 5.27 Result for borrowing rate 30% with link utilization 70% and 30% (10 Mbps)

5.2.1.2.3 Testing With Bandwidth Allocation Of 60% To CL And 40% To BE (10 Mbps)

The next experiment is the Controlled Load service utilizing 60% of the overall link bandwidth and the Best Effort service utilizing the rest of the link bandwidth that is 40%. The borrowing rate for Controlled Load service is set at 10% in the first test, 20% in the second test and 30% in the last test. The FIFO server for Controlled Load service can handle 6000 packets in one second while the FIFO server for Best Effort service can support up to 4000 packets under the 60% and 40% allocation. To simplify the situation, A represents the Controlled load service source while B represents the Best Effort service source.

We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism. Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.29).

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	6000	6000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
2.	CL Light	3000	3000	100	0	0	0
	BE Light	2000	2000	100	0	0	
3.	CL Normal	6000	6000	100	0	0	0
	BE Light	2000	2000	100	0	0	
4.	CL Light	3000	3000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
5.	CL Over	12000	6400	53.33	5600	46.67	23.34
	BE Light	2000	2000	100	0	0	
6.	CL Normal	6000	6000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
7.	CL Light	3000	3000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
8.	CL Over	12000	6000	50	6000	50	25
	BE Normal	4000	4000	100	0	0	
9.	CL Over	12000	6000	50	6000	50	50
	BE Over	8000	4000	50	4000	50	

Table 5.28 Result for borrowing rate 10% with link utilization 60% and 40% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	6000	6000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
2.	CL Light	3000	3000	100	0	0	0
	BE Light	2000	2000	100	0	0	
3.	CL Normal	6000	6000	100	0	0	0
	BE Light	2000	2000	100	0	0	
4.	CL Light	3000	3000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
5.	CL Over	12000	6000	50	6000	50	25
	BE Light	2000	2000	100	0	0	
6.	CL Normal	6000	6000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
7.	CL Light	3000	3000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
8.	CL Over	12000	6000	50	6000	50	25
	BE Normal	4000	4000	100	0	0	
9.	CL Over	12000	6000	50	6000	50	50
	BE Over	8000	4000	50	4000	50	

Table 5.29 Result for link utilization 60% to CL and 40% to BE without borrow mechanism (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	6000	6000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
2.	CL Light	3000	3000	100	0	0	0
	BE Light	2000	2000	100	0	0	
3.	CL Normal	6000	6000	100	0	0	0
	BE Light	2000	2000	100	0	0	
4.	CL Light	3000	3000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
5.	CL Over	12000	6800	56.67	5200	43.33	21.67
	BE Light	2000	2000	100	0	0	
6.	CL Normal	6000	6000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
7.	CL Light	3000	3000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
8.	CL Over	12000	6000	50	6000	50	25
	BE Normal	4000	4000	100	0	0	
9.	CL Over	12000	6000	50	6000	50	50
	BE Over	8000	4000	50	4000	50	

Table 5.30 Result for borrowing rate 20% with link utilization 60% and 40% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	6000	6000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
2.	CL Light	3000	3000	100	0	0	0
	BE Light	2000	2000	100	0	0	
3.	CL Normal	6000	6000	100	0	0	0
	BE Light	2000	2000	100	0	0	
4.	CL Light	3000	3000	100	0	0	0
	BE Normal	4000	4000	100	0	0	
5.	CL Over	12000	7200	60	4800	40	20
	BE Light	2000	2000	100	0	0	
6.	CL Normal	6000	6000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
7.	CL Light	3000	3000	100	0	0	25
	BE Over	8000	4000	50	4000	50	
8.	CL Over	12000	6000	50	6000	50	25
	BE Normal	4000	4000	100	0	0	
9.	CL Over	12000	6000	50	6000	50	50
	BE Over	8000	4000	50	4000	50	

Table 5.31 Result for borrowing rate 30% with link utilization 60% and 40% (10 Mbps)

5.2.1.2.4 Testing With Bandwidth Allocation Of 50% To CL And 50% To BE (10 Mbps)

The last part of this section testing is the Controlled Load service utilizing 50% of the overall link bandwidth and the Best Effort service utilizing the rest of the link bandwidth that is 50%. The borrowing rate for Controlled Load service is set at 10% in the first test, 20% in the second test and 30% in the last test. The FIFO server for Controlled Load service can handle 5000 packets in one second while the FIFO server for Best Effort service can support up to 5000 packets under the 50% and 50% allocation. To simplify the situation, A represents the Controlled load service source while B represents the Best Effort service source.

We also have the experiment without borrow mechanism to compare the result with the experiment with borrow mechanism. Three borrowing rate results i.e 10%, 20% and 30% are compared with the same without borrow mechanism result (table 5.29)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	5000	5000	100	0	0	0
	BE Normal	5000	5000	100	0	0	
2.	CL Light	2500	2500	100	0	0	0
	BE Light	2500	2500	100	0	0	
3.	CL Normal	5000	5000	100	0	0	0
	BE Light	2500	2500	100	0	0	
4.	CL Light	2500	2500	100	0	0	0
	BE Normal	5000	5000	100	0	0	
5.	CL Over	10000	5500	55	4500	45	22.5
	BE Light	2500	2500	100	0	0	
6.	CL Normal	5000	5000	100	0	0	25
	BE Over	10000	5000	50	5000	50	
7.	CL Light	2500	2500	100	0	0	25
	BE Over	10000	5000	50	5000	50	
8.	CL Over	10000	5000	50	5000	50	25
	BE Normal	10000	5000	100	0	0	
9.	CL Over	10000	5000	50	5000	50	50
	BE Over	8000	4000	50	5000	50	

Table 5.32 Result for borrowing rate 10% with link utilization 50% and 50% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	5000	5000	100	0	0	0
	BE Normal	5000	5000	100	0	0	
2.	CL Light	2500	2500	100	0	0	0
	BE Light	2500	2500	100	0	0	
3.	CL Normal	5000	5000	100	0	0	0
	BE Light	2500	2500	100	0	0	
4.	CL Light	2500	2500	100	0	0	0
	BE Normal	5000	5000	100	0	0	
5.	CL Over	10000	5000	50	5000	50	25
	BE Light	2500	2500	100	0	0	
6.	CL Normal	5000	5000	100	0	0	25
	BE Over	10000	5000	50	5000	50	
7.	CL Light	2500	2500	100	0	0	25
	BE Over	10000	5000	50	5000	50	
8.	CL Over	10000	5000	50	5000	50	25
	BE Normal	10000	5000	100	0	0	
9.	CL Over	10000	5000	50	5000	50	50
	BE Over	8000	4000	50	5000	50	

Table 5.33 Result for link utilization 50% to CL and 50% to BE without borrow mechanism (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	5000	5000	100	0	0	0
	BE Normal	5000	5000	100	0	0	
2.	CL Light	2500	2500	100	0	0	0
	BE Light	2500	2500	100	0	0	
3.	CL Normal	5000	5000	100	0	0	0
	BE Light	2500	2500	100	0	0	
4.	CL Light	2500	2500	100	0	0	0
	BE Normal	5000	5000	100	0	0	
5.	CL Over	10000	6000	60	4000	40	20
	BE Light	2500	2500	100	0	0	
6.	CL Normal	5000	5000	100	0	0	25
	BE Over	10000	5000	50	5000	50	
7.	CL Light	2500	2500	100	0	0	25
	BE Over	10000	5000	50	5000	50	
8.	CL Over	10000	5000	50	5000	50	25
	BE Normal	10000	5000	100	0	0	
9.	CL Over	10000	5000	50	5000	50	50
	BE Over	8000	4000	50	5000	50	

Table 5.34 Result for borrowing rate 20% with link utilization 50% and 50% (10 Mbps)

Possibility	Sources situation	Packets generated by the sources	Packets arrive at the destination	Percentage Of Packets Arrive At The Destination (%)	Packets dropped by the destination	Percentage Of Packets Dropped By The Destination (%)	Overall Percentage Of Packets Dropped By The Destination (%)
1.	CL Normal	5000	5000	100	0	0	0
	BE Normal	5000	5000	100	0	0	
2.	CL Light	2500	2500	100	0	0	0
	BE Light	2500	2500	100	0	0	
3.	CL Normal	5000	5000	100	0	0	0
	BE Light	2500	2500	100	0	0	
4.	CL Light	2500	2500	100	0	0	0
	BE Normal	5000	5000	100	0	0	
5.	CL Over	10000	6500	65	3500	35	17.5
	BE Light	2500	2500	100	0	0	
6.	CL Normal	5000	5000	100	0	0	25
	BE Over	10000	5000	50	5000	50	
7.	CL Light	2500	2500	100	0	0	25
	BE Over	10000	5000	50	5000	50	
8.	CL Over	10000	5000	50	5000	50	25
	BE Normal	10000	5000	100	0	0	
9.	CL Over	10000	5000	50	5000	50	50
	BE Over	8000	4000	50	5000	50	

Table 5.35 Result for borrowing rate 30% with link utilization 50% and 50% (10 Mbps)

5.2.1.2.5 Overall Discussion

As mentioned earlier, we only summarize all the discussion at the end of the test because the results and discussion are similar to section 5.2.1.1. There are also do not have any comparison graphs because of the same reason given above.

The interesting result here is that the link-sharing and traffic scheduling algorithm not only successfully provides bandwidth guarantee but if bigger link bandwidth such as 3.0 Mbps or 10 Mbps are used, the numbers of packets dropped in percentage is the same as the numbers of packets dropped if 1.5 Mbps bandwidth is used. Therefore, it is desirable to use bigger bandwidth since this algorithm provides the same quantity of packets dropped in smaller link bandwidth. This is the unique features in the link-sharing and traffic scheduling mechanism.

Others than that, for further details or discussion, one may refer to section 5.2.1..1 we discuss the conclusion and some future enhancement ideas in the next chapter.

Chapter 6

Conclusion And Future Enhancement

6.0 Introduction

Currently, the most commonly used packet forwarding method in the Internet is the First In First Out (FIFO) approach. As explained earlier in Chapter 2, this approach assigns the same priority to all packets and, therefore, cannot provide different levels of service to different data flows. Users cannot request for a certain level of assurance or control in terms of quality of service (QoS) for their traffic.

Other packet scheduling mechanism must be developed to control the sharing of bandwidth on the Internet and, thus, cater to the needs of real-time applications. This project aims to investigate one of the packet scheduling algorithm which attempts to provide bandwidth guarantee i.e the Class Based Queueing (CBQ). This project covers the simulation of the link-sharing and traffic scheduling mechanism to secure bandwidth guarantees to data flows and adopts partially the CBQ features. The rest of this chapter is organized as follows. Section 6.1 presents the objectives and the achievements in this project while section 6.2 gives a brief evaluation of the mechanism being used. We discuss the problems that arised during this project. Finally, some future enhancement methods which gives a few guidelines in understanding and expanding the link-sharing mechanism are presented in the last section.

6.1 Achievements

The main objective of this project is to develop a simulator based on the link-sharing and traffic scheduling mechanism. Besides that, it also observes the performance of the algorithms in guaranteeing the bandwidth using the borrow mechanism under different

situations and link bandwidth utilization. These aspects are important in determining whether link-sharing and traffic scheduling mechanism is a suitable packet scheduling mechanism in the Internet especially for the integrated services.

The overall objectives were achieved and we able to obtain the expected results. We have developed a simulator using the PARSEC simulation language and prove that the link-sharing mechanism is able to provide flow isolation and bandwidth guarantee on our simulation. One of the main features of the CBQ approach that is the "borrow" mechanism able to decrease the number of dropped packets especially during congestion periods. This, of course, reduces the packet loss and, thus, enhances the efficiency of real-time applications.

The important achievements of this project are the capability of the link-sharing mechanism to provide bandwidth guarantee and the ability of the traffic scheduling mechanism to provide lower transit delay to the higher priority flows. This is because the main function of this mechanism is to send packets of higher priority flows first.

Besides that, the link-sharing and traffic mechanism is important in the end-hosts if the transport protocol does not implement any congestion control. Link-sharing mechanism can prevent the monopolies of the bandwidth by certain "misbehaving" application. This can be done by allocating a share of the bandwidth to each class and, therefore, no classes can obtain more than its share of the bandwidth.

Everything in this world have their advantages and disadvantages, therefore this link-sharing and traffic scheduling mechanism are not excluded. Their major drawback by theory is that it do not scale well with the number of flows. In general term, routers at the backbone of Internet often have to route a lot of traffic and in such a situation, this link-sharing and traffic scheduling mechanism would definitely increase the latency of the packets by going through many procedures such as the classification, borrowing and etc. When this mechanism is activated, each packet has to go through the classifier and packet

scheduler, which indirectly increases the time that a packet stays in a node before being forwarded to another router. Furthermore, classification and scheduling a large number of flows demand much computing resources.

6.2 Problems faced during completing the project

There are many problems occurred during developing and completing this project, especially during the compilation of the simulation. During the compilation of this simulator, many errors occurred due to the simulation language we were using is still in the development stage and is not fully commercialized. We have some hard time to find and rectify the problems before it can run. Besides that, we also faced some unknown errors during the runtime such as the segmentation fault [core dumped] and we have to revise our source code many times. Besides that, it was difficult to get information of this topic because not many research have been done on it.

Since almost all the problems above are due to the simulation language, we do not have any specified suggestions to overcome or encounter the problems. We only always seek the advice from our supervisor or the author of this simulation language.

6.3 Evaluation

Finally, we propose that this link-sharing and traffic scheduling mechanism which adopts some of the CBQ features is more suited for providing bandwidth guarantee to aggregated data flows. As the latency of the packets incurred by this mechanism increases with the number of data flows, it is not very suitable for scheduling fine-grained real-time data flows. However, this mechanism can be integrated with other mechanisms such as the inclusion of the admission control or any resources reservation setup to overcome the latency problem.

6.4 Future Enhancement

In this project, we excluded the admission control part and only assumed that all flows are after the admission control setup. In Integrated Service, the perfect traffic control module are provided by the packet scheduler, the packet classifier and admission control. The admission control decides whether QoS requirements of new flows can be met without abusing earlier guarantees. Moreover, admission control can also be employed to enforce administrative policies for QoS arbitration. Hopefully, in the future, this link-sharing and traffic scheduling mechanism can be integrated into the admission control to fully test the traffic control in the integrated service.

Further research may also cover the effect of link-sharing and traffic scheduling on delay jitter and packet loss. The quality of the playback for real-time data is often determined by the delay jitter and packet loss. It is worth having these two aspects for future research in developing more guidelines in understanding and developing the link-sharing mechanism.

REFERENCES

[1] B. Braden, D. Clark, and S. Shenker, Request for Comments (RFC) 1633 ," Integrated Services in the Internet Architecture: An Overview ", IETF, June 1994.

[2] S. Floyd and V. Jacobson, " Link-sharing and resource management models for packet networks ", IEEE/ACM Transactions on Networking, 3(4), 1995. URL http://www-nrg.ee.lbl.gov/nrg-papers.html/

[3] Internet Engineering Task Force (IETF) Integrated Services Working Group (intserv), http://www.ietf.org/html.charters/intserv-charter.html

[4] Internet Engineering Task Force (IETF) Differentiated Services Working Group (diffserv), http://www.ietf.org/html.charters/diffserv-charter.html

[5] Z. Wang, User-Share Differentiation (USD) Scalable bandwidth allocation for differentiated services. Internet Draft, Internet Engineering Task Force, December 1997. Work in progress available at http://www.ietf.org/internet-drafts/draft-wang-diff-serv-usd-00.txt

[6] F. Risso and P. Gevros, " Operational and Performance Issues of a CBQ router ", available from http://www.arciri,org/floyd/papers, October 1999.

[7] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, RFC 2205 , " Resource ReSerVation protocol (RSVP) – Version 1 Functional Specification ", September 1997.

[8] R.Meyer, " PARSEC User Manual Release 1.1 ", available from http://pcl.cs.ucla.edu/, January 1999.

[9] S.Blake, D.Black, M.Carlson, E.Davies, Z.Wang, W.Weiss, " An Architecture for Differentiated Services ", RFC 2475.

[10] J. Heinanen, " Assured Forwarding PHB Group ", Internet Draft, Feb 1999.

[11] V. Jacobson, " Expedited Forwarding PHB ", Internet Draft, Feb 1999.

[12] F. Borgonovo, A.Capone, L.Fratta, C.Petrioli, " VBR bandwidth guaranteed services over DiffServ networks ", August 1999.

[13] Y. Bernet, J.Binder, S.Blake, M.Carlson, S.Keshav, E.Devies, B.Othman, D.Verma, Z.Wang, W.Weiss, " A Framework for Differentiated Services ", Internet Draft, Feb 1999.

[14] S.Floyd, V.Jacobson, " Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, v 1, n4, August 1993.

[15] L. Kleinrock, " Queuing Systems Volume 2 : Computer Applications ", Wiley Interscience, 1975.

[16] P.Ferguson, G.Huston, " Quality of Service in the Internet : Fact, Fiction or Compromise ", INET'98, pp 21-24, July 1998.

[17] S. Keshav, " An Engineering Approach to Computer Networking ", Addison Wesley, pp 234-238, 1997.

[18] A. Demera, S. Keshav, S. Shenker, " Design and Analysis of a Fair Queuing Algorithm ", ACM SIGCOMM'89, Austin, September 1989.

[19] D.Clark, S Shenker, L. Zhang, " Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms ", Proc. SIGCOMM '92, Baltimore, MD, August 1992.

[20] S. Floyd, " Issues in Flexible Resource Management for Datagram Networks ", proceedings of the 3rd Workshop on Very High Speed Networks, March 1992.

[21] V. Jacobson, " Private Communication ", 1991.

[22] R. Braden, D.Clark, S.Shenker, " Integrated Services in the Internet Architecture : an Overview ", Internet Draft, June 1995.

[23] S. Schenker, C. Partridge and R. Guerin, " Specification of Guaranteed Quality of Service," Internet Draft, Aug 1996, ftp://ds.internic.net/internet-drafts/draft-ietf-intserv-guaranteed-svc-06.txt.

[24] P.White, " RSVP and Integrated Services in the Internet : A Tutorial ", IEEE Communications Magazine, May 1997.

[25] A.Parekh, R. Gallagher, " A Generalized processor Sharing Approach to Flow Control – the Single Node Case,", IEEE/ACM Trans Networking, vol. 1, no 3, 1993, pp 366-57.

[26] A.Parekh, R. Gallagher, " A Generalized processor Sharing Approach to Flow Control – the Multiple Node Case,", IEEE/ACM Trans Networking, vo2. 1, no 2, 1996, pp 137-50.

[27] J. Wroclawski, " Specification of the Controlled-Load Network Element Service ", Internet Draft, Aug 1996, ftp://ds.internic.net/internet-drafts/draft-ietf-intserv-ctrl-load-svc-03.txt.

[28] S. Floyd and M. Speer, Lbnl's cbq code v2.0, May 1997, ftp://ftp.ee/lblb.gov/cbq2.0.tar.Z.

[29] S. Floyd, " Notes on the relationship between CBQ and RSVP ", 1 October 1997, http://www.aciri.org/floyd/RS.

[30] " Parallel Queuing Network Simulation ", pqnsim. Html, available from http://pcl.cs.ucla.edu/projects/parsec/pqn-sim/

[31] C. Casetti, J. Kurose, D. Towsley. " An Adaptive Algorithm for Measuement-based Admission Control to Integrated Services Packet Networks ", Techincal Report TR 96 –76, 1996.