# A Selective Approach to Bandwidth Overbooking

Feng Huang
*Brigham Young University - Provo*

# A SELECTIVE APPROACH TO BANDWIDTH OVERBOOKING

by

Feng Huang

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment for the degree of

Master of Science

Department of Computer Science

Brigham Young University

April 2006

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Feng Huang

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

| | |
|---|---|
| Date | Mark J. Clement, Committee Chairman |
| Date | Quinn O. Snell, Committee Member |
| Date | Irene Langkilde-Geary, Committee Member |

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Feng Huang in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

| | |
|---|---|
| ______________________ | ______________________ |
| Date | Mark J. Clement,<br>Chair, Graduate Committee |
| Accepted for the Department | ______________________<br>Parris Egbert<br>Graduate Coordinator |
| Accepted for the College | ______________________<br>Thomas W. Sederberg<br>Associate Dean, College of Physical and Mathematical Science |

## ABSTRACT

## A SELECTIVE APPROACH TO BANDWIDTH OVERBOOKING

Feng Huang

Department of Computer Science

Master of Science

Overbooking is a technique used by network providers to increase bandwidth utilization. If the overbooking factor is chosen appropriately, additional virtual circuits can be admitted without degrading quality of service for existing customers. Most existing implementations use a single factor to accept a linear fraction of traffic requests. High values of this factor may cause the degradation of quality of service whereas low overbooking factors will result in underutilization of bandwidth. Network providers often select overbooking factors based only on aggregate average virtual circuit utilization. This paper proposes a selective overbooking scheme based on trunk size and usage profile. Experiments and analysis show that the new overbooking policy results in a superior network performance.

## ACKNOWLEDGMENTS

I would like to thank Dr. Clement for his intelligence leadership and encouragement during this research.

I am also thankful to Dr. Snell and Dr. Geary for their advice on my research and comments on this thesis.

I would like to thank Casey Deccio, Ian Garcia and other members in Network Computing Laboratory for their invaluable help.

I thank my parents and sister for their support and encouragement.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF EQUATIONS

# Chapter 1 Introduction

The demand for network bandwidth has increased significantly as internet-related traffic has grown during the last decade. In addition to installing more expensive physical network links, network management can also play an important role in accommodating this increased traffic. Bandwidth overbooking can be used to virtually increase available bandwidth and improve network efficiency. If the overbooking factor is chosen appropriately, additional virtual circuits (VCs) can be admitted without degrading quality of service for existing customers. This work proposes a flexible overbooking scheme and analyzes its impact on network performance.

## 1.1 Bandwidth Overbooking

Overbooking is a term used to describe the extra sale of network transport access. When bandwidth overbooking is employed, each source (network user) which has traffic admitted to a backbone link is assigned less bandwidth than they request. This leads to more admitted VCs, but the sum of requested bandwidth is greater than the trunk capacity. As long as the overbooking factor is chosen to correctly predict actual VC utilization, overbooking results in higher profit margins for network service providers and it will also benefit the users with lower cost of subscription.

Customers often request more bandwidth than they require. This leads to low utilization, particularly when a large number of circuits are aggregated on a backbone link. One common misconception is that the Internet is always congested. Although some parts of the Internet, such as university networks, are highly utilized, the backbones of the Internet are relatively lightly loaded [ODL00]. This underutilized resource makes overbooking possible.

The Internet uses statistical multiplexing to aggregate traffic. In some literature, the overbooking ratio is equivalent to the term "statistical multiplexing gain." When statistical multiplexing is used, all user data packets are assigned to shared links. Unlike time division multiplexing (TDM) and frequency division multiplexing (FDM), statistical multiplexing does not reserve bandwidth for each user. Since most users send bursty traffic, statistical multiplexing allows active circuits to use bandwidth during the idle time of other circuits. When a constant bit rate (CBR) VC is admitted into a link, network administrators must allocate bandwidth equal to its peak rate. When variable bit rate (VBR) or available bit rate (ABR) VCs are admitted, they do not need bandwidth equal to the sum of their peak rates. The probability that all users will send traffic at the peak rate at the same time is extremely low. Since users generate traffic independently on the Internet, their traffic peaks and idle periods should occur at different times. The probability that one stream will be able to use spare capacity from another stream increases when there are a large number of users in the network.

Overbooking has been used in data networks and other services. Airlines usually sell more seats than are physically available because not all the consumers use their tickets. If the policy sells too many additional tickets, there will be some passengers that have paid for a ticket who will not be able to use that flight. Airlines will typically offer free flights at another time in order to compensate customers for the inconvenience of taking another flight. If this lack of "Quality of Service" inflight scheduling becomes too severe, customers may select another airline and the profit margin will decrease because of decreased demand. If the policy overestimates the number of passengers that will show up for a given flight, then seats will go empty and revenue will be lost. Like airline companies, network service providers have to be careful in selecting overbooking factors. Network administrators can maximize profit by accurately predicting the actual utilization of VCs in order to specify an overbooking policy that will sell as much bandwidth as possible without causing degradation in network "Quality of Service" (QoS). Poor "Quality of Service" results in dropped packets and increased delay and jitter, which affect the customers' satisfaction.

## 1.2 The General Picture of the Bandwidth Market

The networking infrastructure consists of backbone providers, Internet Service Providers (ISP), and users. ISPs sell bandwidth to users and buy from backbone providers. Backbone suppliers own high bandwidth trunk links and may also directly sell bandwidth to large users. Individual users buy bandwidth from ISPs through various types of contracts. They are often fix-priced based on the peak amount of bandwidth they require even though the actual usage may be lower. The Committed Information Rate (CIR) is the requested bandwidth from the users. The Peak Information Rate (PIR) is the peak rate possible for the circuit. Given a set of requests containing both CIR and PIR values, an ISP can decide what percentage of the CIR to reserve for each request. This research is focused on this overbooking policy used by ISPs.

Currently there is no standard overbooking policy for network suppliers. Policies vary depending on the service provider and public network. For example, telephone networks are built to service 5% of subscribers simultaneously. ISPs typically use a 10:1 modem ratio. DSL home service is overbooked at 50:1 and business DSL at 20:1. Web hosting generally uses 4:1 overbooking on space and 10:1 overbooking on bandwidth. It is very common for 20% or less of the customers to use 80% of a server's resources [SAL03].

Overbooking bandwidth on an internet router is standard practice. Most modern routers are implemented with overbooking functionality. For example, ATM routers can overbook VBR and ABR traffic. Overbooking is not performed with CBR traffic because the traffic rate is fixed. Routers have implemented per-link and per-class overbooking using local overbooking multipliers [CIS04].

## 1.3 Network Traffic Models

Data traffic patterns have significant implications for network performance. Poisson traffic models have been tremendously useful in designing and analyzing networks. However it has been found that the Poisson traffic model is not suitable for all network traffic. In some environments, self-similar traffic is shown to violate the Poisson model [LEL94] [PAX95].

The Poisson-based model originated in telecommunication voice networks. This model assumes a Poisson arrival process and Poisson call duration. The Poisson process assumption is the basis for well established queuing theory. Voice traffic conforms to Poisson models.

Self-similar patterns repeat at different spatial or time scales. Self-similar patterns exist extensively in nature. The shapes of leaves, rivers, etc. all show self-similarity. Leland et al have shown that Ethernet traffic exhibits self-similar properties [LEL94]. Other research affirms that web traffic is also self-similar [CRO97]. Self-similar traffic does not aggregate as smoothly as Poisson traffic.

Poisson traffic and self-similar traffic have a different effect on overbooking strategies. Poisson traffic will be smoothed as the number of users increases but self-similar traffic will still remain relatively bursty. This difference is shown in Figure 1. In general, self-similar traffic is more difficult to overbook. This research uses real traffic traces to capture characteristics that would not be observed using Poisson assumptions.

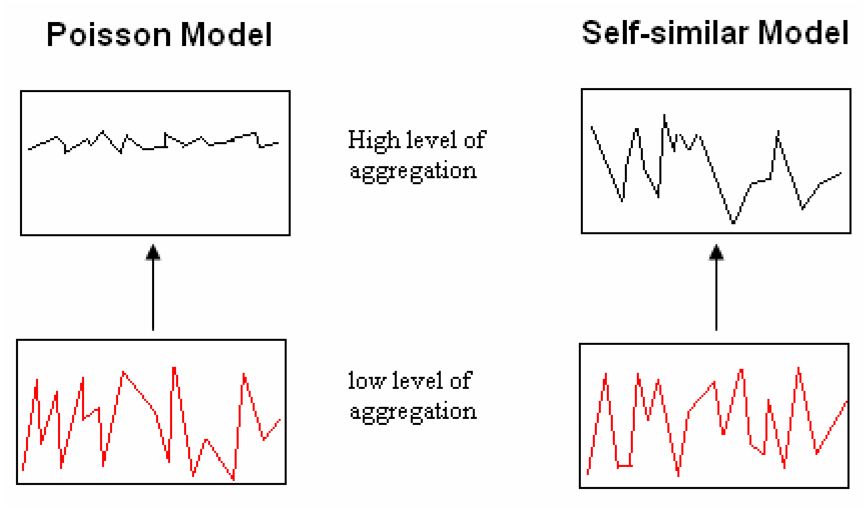


Figure 1.1: Poisson and self-similar traffic. Poisson traffic tends to be smoother after aggregating whereas the self-similar traffic remains bursty.

## 1.4 Thesis Statement

Enhanced overbooking techniques can increase the number of admitted flows while maintaining QoS. The overbooking factor should provide balance between economic considerations and performance objectives such as delay and packet loss. When trunk type, network traffic type and customer characteristics are considered in setting overbooking factors, bandwidth can be used more efficiently and utilization can be predicted more accurately.

## 1.5 Research Approach

This research analyzes bandwidth overbooking for network service providers. Real backbone network traffic is used to discover traffic usage patterns. This research proposes a selective approach for bandwidth overbooking based on usage patterns and trunk type. Its impact on network performance is examined through simulation experiments.

The remaining chapters are organized as follow. Chapter 2 presents an analysis of traffic contract parameters and trunk sizes. Chapter 3 proposes a bandwidth utilization performance criterion for overbooking policies. Chapter 4 shows the experimental results of overbooking based on usage profiles. Chapter 5 shows the experimental results of overbooking based on trunk size. Chapter 6 concludes and explores future work.

# Chapter 2 Traffic Data Analysis

Bandwidth usage patterns have direct implications for overbooking policy design. This chapter presents the results of the statistical analysis of backbone traffic data, which will be the basis for the overbooking scheme in this research.

## 2.1 Introduction

Bandwidth utilization is one of the means by which network service providers determine the quality of networks. These utilization values decide the cost of service since transmission links are currently priced by their maximal capacity. This research examines the utilization of VCs and backbone links and provides insight into usage patterns.

The overbooking factor is multiplied by the bandwidth request (CIR) from the customer in order to determine actual bandwidth allocated for its virtual circuit. A smaller value of this factor means more aggressive or higher overbooking. For example, when the overbooking value is 0.2 (20%), a customer requesting a T1 (1.5Kbps) line is only allocated 0.3 Kbps. When the value is 0.6 (60%), the allocated bandwidth is 0.9kpbs for the same request. The overbooking value of 0.2 can accept more VCs for trunks and is more aggressive than the one of 0.6.

There is a large variance in network usage for different users. Corporation circuits have light utilization of about 1% whereas private home links are heavily used [ODL00]. The statistical analysis in the following sections shows that users in different CIR classes also exhibit differences.

This research analyzes data from up to 475,230 Frame Relay virtual circuits on a wide area network with approximately 2000 trunks. Data were retrieved during normal work weeks in 2002 and 2003. The network is similar in nature to backbones supported by many national providers. This research derives the relationship between network variables and utilization. These variables include CIR, PIR and link size. The cost of a virtual circuit is based on the average of CIR. Traffic above PIR is marked and will be dropped when backbone trunks get congested. The results of the statistical analysis shows some interesting patterns in the traffic data from AT&T backbone links that suggests that a piecewise linear overbooking policy could be beneficial.

## 2.2 Utilization Based on Contract Parameters

A summary of VC utilization is shown in Table 2.1. The PIR may be much larger than the CIR and the customer should be able to send at rates between CIR and PIR for short periods of time as long as the average bandwidth is no greater than the CIR. The actual bandwidth utilization is shown in the table along with the number of virtual circuits that had that average utilization.  The total values are weighted by the magnitude of the bandwidth so that large VCs have a proportionally more significant impact on total averages than smaller VCs. The average utilization across all 400,000 virtual circuits was approximately 28%. This means that a provider could admit nearly four times as much as traffic with an overbooking policy than he could with a strict admission policy. This overbooking should not result in lower quality of service since users are underutilizing their links by this factor.

| CIR Range | PIR Range (Kbps) | | | | |
|---|---|---|---|---|---|
| (Kbps) | 0 512 | 513 1536 | 1537 2212 | 2213 + | Totals: |
| 0 5 | 36.48% (49330) | **109.06%** (2930) | 99.46% (9830) | **192.77%** (440) | 53.56% (62530) |
| 6 64 | 27.16% (193360) | 39.06% (24090) | 34.13% (56050) | 27.02% (690) | 29.24% (274190) |
| 65 128 | 27.77% (28390) | 34.91% (10940) | 28.10% (17040) | 32.90% (290) | 29.18% (56660) |
| 129 256 | 30.07% (4970) | 30.98% (17570) | 29.31% (20250) | 72.24% (440) | 30.61% (43230) |
| 257 512 | 42.21% (50) | 28.74% (8910) | 29.42% (12810) | 30.40% (660) | 29.16% (22430) |
| 513 1024 | 0.00% (0) | 27.00% (1690) | 27.34% (8520) | 26.48% (670) | 27.22% (10880) |
| 1025 + | 0.00% (0) | 14.44% (310) | 20.67% (3550) | 28.40% (1000) | 23.62% (5310) |
| Totals: | 27.82% (276100) | 29.97% (66440) | 27.13% (128050) | 29.40% (4190) | 28.15% (475230) |

Table 2.1 Utilization values for ranges of CIR and PIR values. The numbers in parentheses are the total number of virtual circuits in that range.

Table 2.1 also shows that there are some users that significantly overutilize their VCs. Note that a significant number of VCs use as high as 193% of the negotiated CIR. A successful overbooking policy must correctly estimate utilization for both small and large VCs. A piecewise linear approach is well suited to predict this utilization.

Several techniques have been explored to determine accurate overbooking techniques.

The most obvious prediction technique is to use the CIR (the customers' best estimate of bandwidth usage) multiplied by some constant to allocate bandwidth. Intuitively, it may seem likely that the PIR could also be combined in a linear system with two variables.

A statistical regression analysis was performed to determine the correlation between CIR, PIR and utilization. Results shown in Table 2.2 indicate that CIR is significant in predicting VC utilization. The linear model also indicates an overbooking factor of 20.4% based on the linear fit to the data. The PIR is also correlated with utilization, but explains less of the variance in the data than CIR. The mean squared error did not decrease significantly when using PIR and CIR over CIR alone. This also shows that the inclusion of PIR in an overbooking technique may not lead to more accurate predictions of utilization.

| **Estimate** | | **MS Error** | **t value** | | **Pr > \|t\|** | |
|---|---|---|---|---|---|---|
| CIR alone | | | | | | |
| 0.2115 | | 9398 | 421 | | <.0001 | |
| CIR and PIR | | | | | | |
| CIR | PIR | | CIR | PIR | CIR | PIR |
| 0.204 | 0.004 | 9368 | 304 | 39 | <.0001 | <.0001 |

Table 2.2. GLM regression analysis of CIR and PIR correlation with utilization. The first set of results is from a regression using only CIR. The second set results from a linear regression with both PIR and CIR.

A more detailed view of the relationship between CIR and utilization is shown in Figure 2.1 and Figure 2.2. They are for VC data from 2002 and 2003 respectively. Some CIR values are associated with a small number of VCs and those VCs are not included in order to reduce the effect of outliers. Data set 1 is for 2002 and consists of 30 classes of

CIR values. All CIRs shown in Figure 2.1 have greater than 500 VCs. Data set 2 from 2003 is more diverse and consists of 160 classes of CIRs. All CIRs shown in Figure 2.2 have more than 30 VCs.

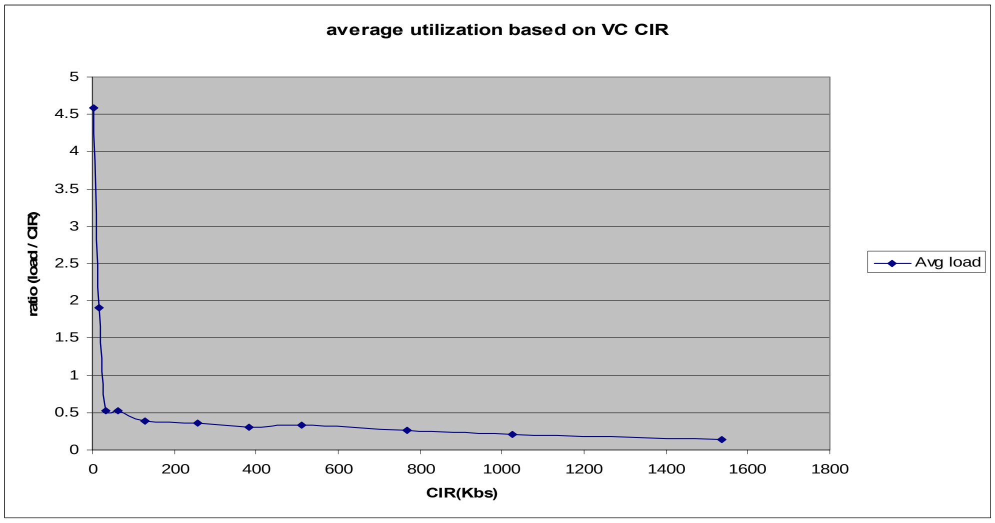


Figure 2.1. Utilization shows a negative correlation with CIR for data set 2002. The y-axis represents the average utilization whereas the x-axis is CIR value.

In Figure 2.1, utilization of VCs decreases as CIR values increase. CIR values range from 1 to 1536kbs. For VCs with CIR of 4kbs, the average utilization is as high as 458%. On the other hand, for VCs with CIR of 1536kbs, the utilization is only 14%. This strongly suggests a high overbooking factor for large VCs and low overbooking for small VCs.

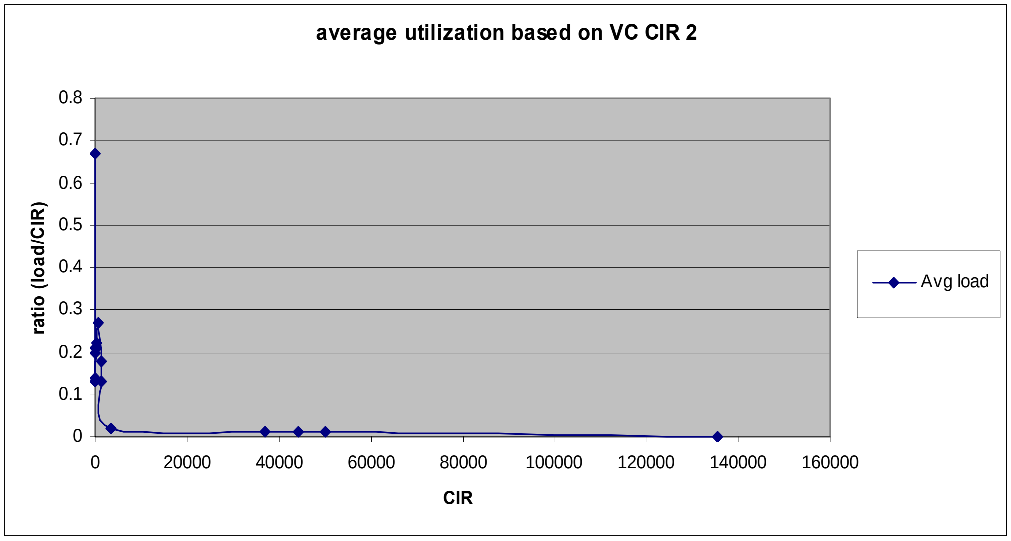


Figure 2.2 Utilization shows a negative correlation with CIR for data set 2003. The y-axis represents the average utilization whereas the x-axis is CIR value.

Figure 2.2 also shows a negative correlation between CIR and utilization for 2003 data. This set of data is much more diverse; CIR values range from 5 to 135,631 kbs. There are three distinct regions for this set of data. For VCs with CIR of smaller than 16 kbs, the average utilization is above 70%. For VCs with CIR of larger than 40,000 kbs, the average utilization is lower than 2%. All other VCs have an average utilization between 10% and 70%.

## 2.3 Utilization Based on Trunk Size

Hardware vendors have recently incorporated additional overbooking hardware in core switches to allow for unique linear overbooking factors to be applied on a trunk by trunk basis. This new option is useful if there is a relationship between utilization and trunk size. Data from three types of trunks, T3 (96000 cps), STS1 (104150 cps) and OC3

(353207 cps), are analyzed. The multiple regression result is shown in Figure 2.3.

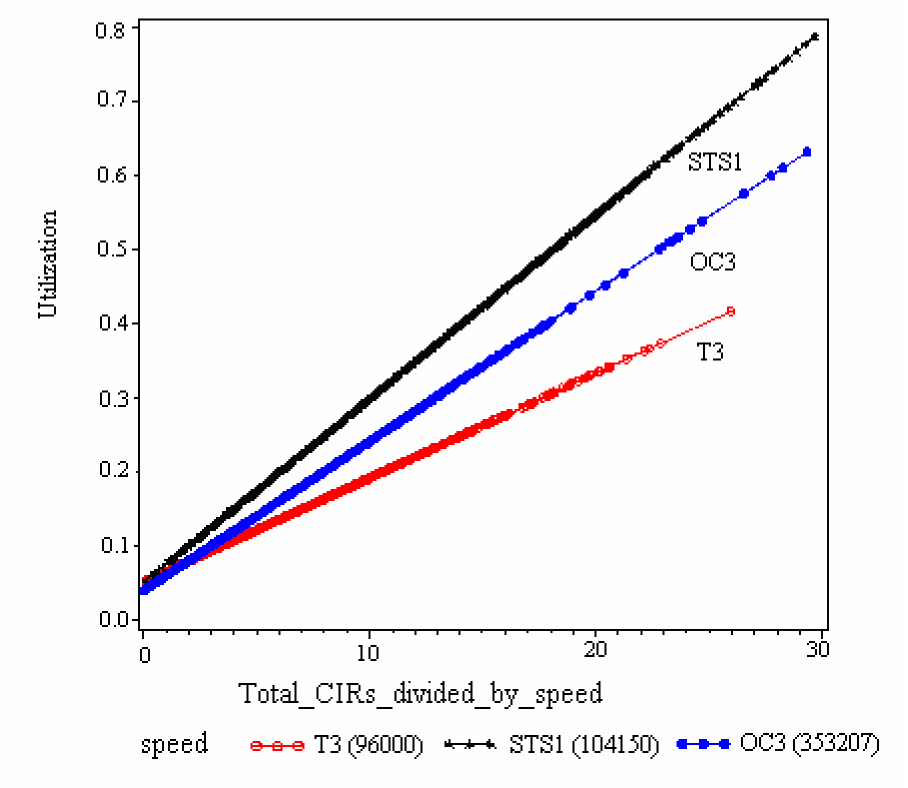


Figure 2.3 Predicted utilization based on trunk size and total CIRs. The y-axis represents the predicted utilization and the x-axis represents the value of total CIRs divided by trunk bandwidth (speed).

In Figure 2.3, utilization does not show a significant correlation with trunk size. T3 links are the least utilized among the three whereas STS1 links appear to be used most aggressively. This may result from different numbers of VCs in trunks.

The simple regression analysis above derives the linear relationship between the total CIR and utilization. However it does not give a clear picture of traffic characteristics for individual trunks. In Figure 2.4, single regression analysis shows the relationship between utilization and total CIR (overbooking factor) as non-linear for STS1 trunks. The traffic from T3 and OC3 trunks show similar patterns. The utilization increases linearly with the value of total CIR when CIR is small compared to trunk bandwidth. However, when the value of total CIR becomes larger, the level of increase slows down and utilization stabilizes at around 40%. Consumers who purchase larger CIRs are not as aggressive in using bandwidth as smaller customers. This conservative usage suggests that higher overbooking factors could be adopted for these high CIR customers.

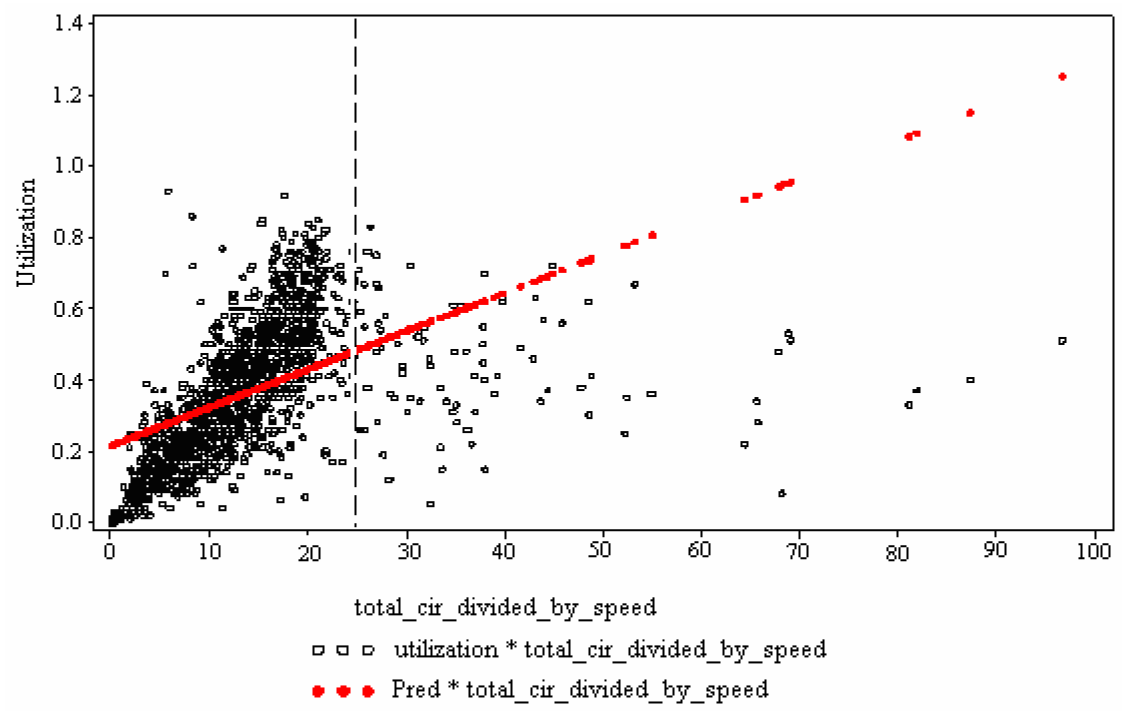


Figure 2.4 The single regress analysis for STS1 (104000 cps) trunks. The y-axis plots the actual utilization for each STS1 trunk. The x-axis represents the value of total CIRs divided by trunk bandwidth (speed). The dotted straight line is the predicated utilization based on regression analysis.

As shown in Figure 2.4, the traffic pattern for STS1 trunks has two distinct regions for the different values of total CIRs divided by trunk bandwidth (speed). When these values are greater than 25, trunks are less utilized and utilization is around 40%. For those trunks, higher overbooking values could be used to improve utilization. When the values of total CIRs divided by speed are less than 25, the utilization increases quickly to 90%.

For those trunks, more overbooking will result in the degradation of QoS. This suggests that two different levels of overbooking should be used for STS1 trunks.

## 2.4 Summary

Statistical analysis for backbone traffic data shows a variety of usage patterns for different users. This provides a statistical basis for selective overbooking strategies. The piecewise linear overbooking will result in better network performance.

# Chapter 3 Performance Criteria

Network providers need to have specific performance objectives for their network services. Those objectives must be balanced between economic factors and quality of service. This chapter first examines common performance metrics in networking. Then the optimal performance criteria are derived from queuing theory. Finally, performance objectives for the overbooking policy in this research are validated.

## 3.1 Network Performance Metrics

Throughput and delay are two significant metrics for any networking system. Throughput is data transfer rate measured in bits per second. Delay corresponds to the amount of time it takes for a packet to travel from source to destination. Delay is usually composed of propagation delay, transmit delay and queue delay. Jitters and packet loss rate are also important metrics. Jitter is the measurement of delay variance. Packet loss rate is defined as the percentage of packets lost during transmission generally due to buffer overflow. Large values of delay and packet loss indicate network congestion.

Network applications may have different performance requirements. File transfer and email are throughput sensitive whereas interactive applications, such as Voice over IP (VoIP), have stricter delay and jitter constrains.

Network managers and administrators want stable network performance as well as maximal resource utilization. Full link utilization seems to lead to a maximal usage of bandwidth. However, high utilization and heavy load may cause high values of delay and packet loss. In choosing an overbooking policy, link utilization is the primary performance goal whereas delay and packet loss rate should meet the requirements of most applications. The following sections discuss the ideal network performance as well as practical considerations for utilization.

## 3.2 Optimization Criteria

Offered load affects both delay and throughput. Figure3.1a shows the general patterns of these two metrics as a function of load [JAI88]. The throughput increases until the load approaches network capacity. Throughput suddenly drops at this point as queues overflow, causing packets to be dropped. This is the point referred to as the throughput 'cliff' point, where severe network congestion occurs. Delay increases linearly until the buffer begins to build up and then increases exponentially.

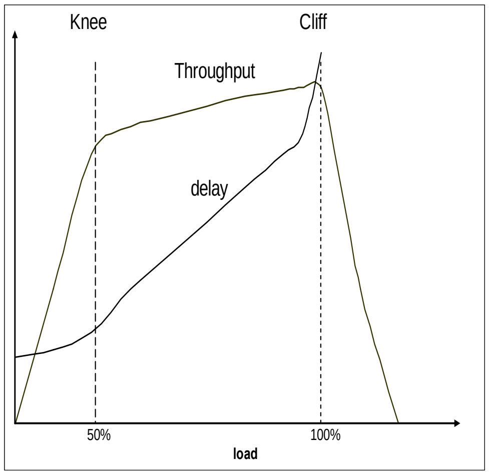


Figure3.1a: Throughput and delay curves for a M/M/1 queue as a function of load

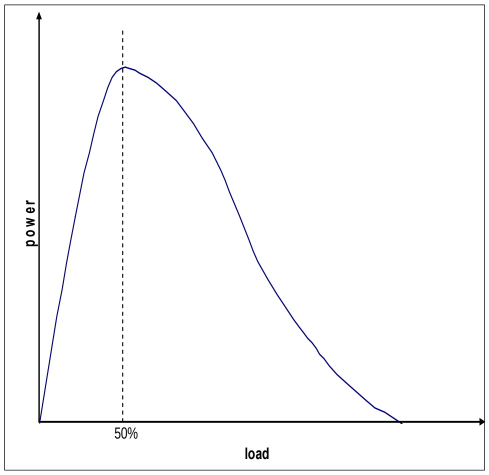


Figure3.1b: Power as a function of load for a M/M/1 queue

Delay and throughput are related to each other and they appear to be redundant when used to measure network performance. The power metric is defined as a combination of these two metrics.

The power metric is the ratio of throughput to delay. Research has shown that power has a single maximum [GIE78] [KLE79]. This maximal point is proposed as the optimal operating point at the 'knee' of throughput and delay curve as shown in Figure 3.1b. At this point, throughput is relatively high whereas delay is still increasing only gradually. The maximal power can be solved mathematically using queuing theory.

### 3.2.1 Queuing Theory

Queuing theory plays a key role in modeling and analyzing networks. It is used to determine the statistics of a queue from which desired performance metrics, such as queue length or loss probability, may be derived. Combined with *Little's formula* [LIT61], queuing theory can also calculate queue size from queue delay.

A common queue representation appears in Figure 3.2. The relevant parameters are arrival rate or load $\lambda$, link capacity $\mu$ and queue length $n$. Link utilization $\rho$ is the ratio of load to capacity when load is less than capacity.

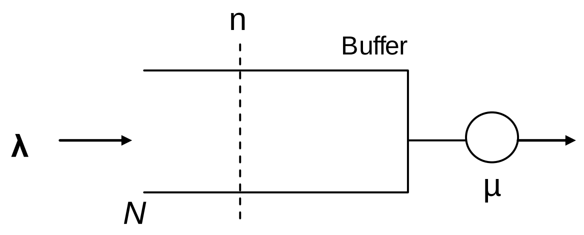


Figure 3.2: Representation of queue

Considering an infinite M/M/1 queue, the average queue size is

$$E(n) = \sum_{n=0}^{\infty} np_n = \frac{\rho}{1-\rho} \tag{3-1}$$

The average size increases as ρ approaches 1. For ρ < 0.5 the average number of packets in the queue is less than 1. For ρ = 0.8, E (n) = 4.

Applying *Little's formula* [LIT61] to equation (3-1), the average queue delay becomes

$$E(T) = \frac{1}{\mu(1-\rho)} = \frac{1}{\mu-\lambda} \tag{3-2}$$

Since the throughput is the load λ for an infinite queue, power is given by

$$\begin{aligned} \text{Power}|_{M/M/1} &= \lambda / E(T) \\ &= \lambda\mu(1-\rho) \\ &= \rho\mu^2(1-\rho) \end{aligned} \tag{3-3}$$

The maximal power value can be found by differentiating equation (3-3) with respect to ρ and setting it to zero.

$$\begin{aligned}
&\rho\mu^2(1-\rho)\\
&\frac{d}{d\rho}\rho\mu^2(1-\rho)\\
&=\mu^2-2\mu^2\rho\\
&=\mu^2(1-2\rho)
\end{aligned}$$

It yields $\mu^2(1-2\rho)=0$, then

$$\begin{aligned}
&\mu^2=2\rho\mu^2\\
&1=2\rho\\
&\rho=1/2
\end{aligned}$$

This process shows that the maximum is $0.25\mu^2$ when utilization is 0.5. For an infinite M/M/1 queue, the optimal performance point exists where link bandwidth is half used. An interesting result is that the queue size is 1 for this case. This means that there is only one packet waiting for service at the point of maximum power. Similar analysis leads to utilization of 0.586 for an infinite M/D/1 queue.

## 3.3 Optimal Criteria for Overbooking

Overbooking is used to increase bandwidth utilization. Profit margin is the driving force behind it. The 50% utilization for an M/M/1 queue is too low to satisfy the intention of network providers. Noted in Figure 3.1, the optimal point happens at the 'knee' of throughput and delay curves. There is still potential for improvement before the "cliff" point, where congestion is severe. This suggests that a heavier load may be chosen for acceptable network performance.

Take 80% utilization as an example. Throughput is reasonably high for this case. The following calculations show that this utilization value will also satisfy delay and loss rate requirements for most network applications.

Queue delay can be calculated using equation (3-2). In this equation, delay is inversely proportional to link capacity. Consider a T1 link with 15Mbs as the worst case. When utilization reaches 0.8, the queue delay is 2 ms from equation (3-2). Normally, delay within 150ms is acceptable for interactive applications. Since propagation delay for backbone networks is less than 50ms, this queue delay should be low enough for most interactive applications.

For packet loss rate, consider equation (3-4). This equation shows the probability that the queue exceeds a specified number [MIS96].

$$P(n > N) = \sum_{n=N+1}^{\infty} p_n = \rho^{N+1} \tag{3-4}$$

The average queue size is 4 for $\rho = 0.8$ as indicated in previous section. From equation (3-4), the chance of exceeding 30 packets is less than $1.2 \times 10^{-3}$. Backbone networks will have more than 30 buffers [CIS05] [ROM94]. So this value of $\rho$ will satisfy most requirements for loss rate.

Higher utilization leads to longer queue sizes. A utilization of 0.9 will need a buffer size of 60 for a loss probability of $1 \times 10^{-3}$. This doubles the buffer size required for a utilization value of 0.8

### 3.3.1 Analysis for Self-similar Traffic

Self-similar traffic is burstier than Poisson traffic. This causes more packet loss and longer queue delay for self-similar traffic given the same average load. Additional resources, such as buffer space, are needed for self-similar traffic if similar performance quality is required. This subsection derives queue performance results for general self-similar traffic.

Norros developed a workload model based on fractional Brownian motion (FBM) [NOR95]. The following equation can be used to calculate the queue size for self-similar traffic based on FBM.

$$q = \frac{\rho^{1/2(1-H)}}{(1-\rho)^{H/(1-H)}} \tag{3-5}$$

In this equation, *H* is the Hurst parameter, which indicates the burstiness of self-similar traffic. When *H* has values between 0.7 and 0.9, traffic shows self-similarity.

Let *H* = 0.75, a common value for self-similar traffic for this calculation [WIL98]. When ρ is equal to 0.8, queue size and queue delay are 80 and 6.7ms respectively. Those values will still satisfy most requirements for network applications.

### 3.3.2 Real Time Considerations

The performance criteria of a network are the responsibility of network managers. They may choose to maximize throughput at the cost of quality of service, or they may keep a low utilization to satisfy some application with extremely strict delay requirements. For the overbooking policy in this research, a utilization of 80% is chosen as the target utilization.

## 3.4 Summary

Using queuing theory, the optimal operating point for power function is derived. A higher utilization standard can be adopted for the purpose of overbooking. Considering realistic delay and loss rate requirements, a target utilization of 80% is chosen in this research. In next sections, the results of simulation experiments under these performance criteria will be discussed.

# Chapter 4 Trunk Based Overbooking

## 4.1 Introduction

It is reasonable to believe that large trunks may be able to benefit from increased statistical multiplexing so that a more aggressive overbooking factor could be used. Hardware vendors have incorporated additional overbooking hardware in core switches to allow for an increased linear factor to be applied on a trunk by trunk basis. Theoretical analysis and simulation experiments show that larger overbooking values can be used for large trunks than for small trunks.

## 4.2 Network Topology and Simulation Setting

Simulation is an important tool in network analysis [LEE99] [FLO01]. Compared to small-scale evaluation in a lab or wide-area test beds, the simulation is much less expensive and easier to repeat. The *ns-2* simulator is used in this research. It was developed at UC Berkeley and widely used to simulate large-scale networks [NS06].

A dumbbell topology [Figure 4.1] is set up to simulate the traffic on a network backbone. Thousands of flows are connected to the ends of this backbone. The actual traffic values are derived from the AT&T backbone traffic data. The network performance results are retrieved through the queue monitor at the egress node of backbone. Based on the different settings of network parameters in the ns program, the simulation can test various traffic types and overbooking schemes.

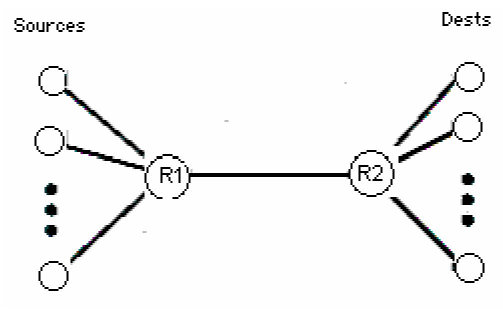


Figure 4.1: Dumbbell topology for network simulation. Different traffic setting can be used for overbooking test. Performance is monitored by queue monitor on R2.

Exponential sources were configured in the *ns* simulator with average bandwidth determined by the utilization value for each source randomly chosen from the AT&T trace file. Experiments were performed with each overbooking factor to determine the impact of the overbooking policy on utilization and packet loss. To simulate the real time internet traffic, 90% of the traffic is TCP traffic and 10% is UDP in most simulations.

## 4.3 Effective Bandwidth

Effective bandwidth more accurately describes the resource usage for a source within a link. It is a summary of statistical characteristics of sources over different time scales and buffer sizes. Effective bandwidth provides a better estimation of resource usage than a simple count of the bits carried. For example, bursty traffic may require low utilization to meet tight delay requirements. On the other hand, constant rate traffic may meet delay requirements with much higher utilization values.

The following effective bandwidth equation is widely accepted in the field [KEL96].

$$\alpha(s,t) = \frac{1}{st} \log E\left[\exp(sX[0,t])\right] \qquad (4\text{-}1)$$

X[0,t] is the amount of workload produced by a source in the time interval of length *t*. *E*[exp(*sX*[0,t]] denotes the expected value of the exponential of *s* times this workload. The two most important parameters for effective bandwidth are the space parameter '*s*' and the time parameter '*t*'. These two parameters characterize the context of the source such as the level of multiplexing and overflow probability [SIR00].

The parameter *s* is an indication of the degree of statistical multiplexing. When the link capacity is much larger than the peak rate of the multiplexing sources, this parameter has a small value. Conversely, when the peak rate of the source is near the link capacity, there is a low degree of multiplexing and large values of *s*. Infinite values of *s* correspond to deterministic multiplexing. The parameter *s* is in units of $kbyte^{-1}$.

The parameter *t* corresponds to the most probable duration of the buffer busy period prior to overflow. It is the indication of the time scales related to buffer overflow. It is measured in units of msecs or seconds depending on the buffer size.

### 4.3.1 Aggregation Effect on Link Capacity

Theoretical effective bandwidth of a source decreases as the level of statistical multiplexing increases in the link. Research has shown experiment results for compressed MPEG traffic [CSS99]. With a mean rate of 26mbps for traffic streams, the effective bandwidth of a single stream is EB = 0.54, 0.33, 0.26 for three link with capacity of 34, 155, and 622 Mbps respectively. The effective bandwidth of the MPEG source is greater for smaller trunks.

Intuitively, large link capacity has more space for increased statistical multiplexing because large trunks may accept more traffic sources. When the number of independent sources on a link increases, the odds that the sources send traffic simultaneously decreases. This gives large links higher potential for overbooking than small links under the similar QoS constraints.

## 4.4 Simulation Experiments

To test the performance difference for trunk sizes, similar simulation environments were setup for different trunks. All parameters were the same except for trunk size. Experimental results show that larger trunks may use larger overbooking values.

Figure 4.2 shows the average utilization for two trunks with the same overbooking factors. They have similar average utilization for most overbooking values. This is reasonable because both trunks have similar ratios of admitted traffic load to trunk size with the same overbooking factor.

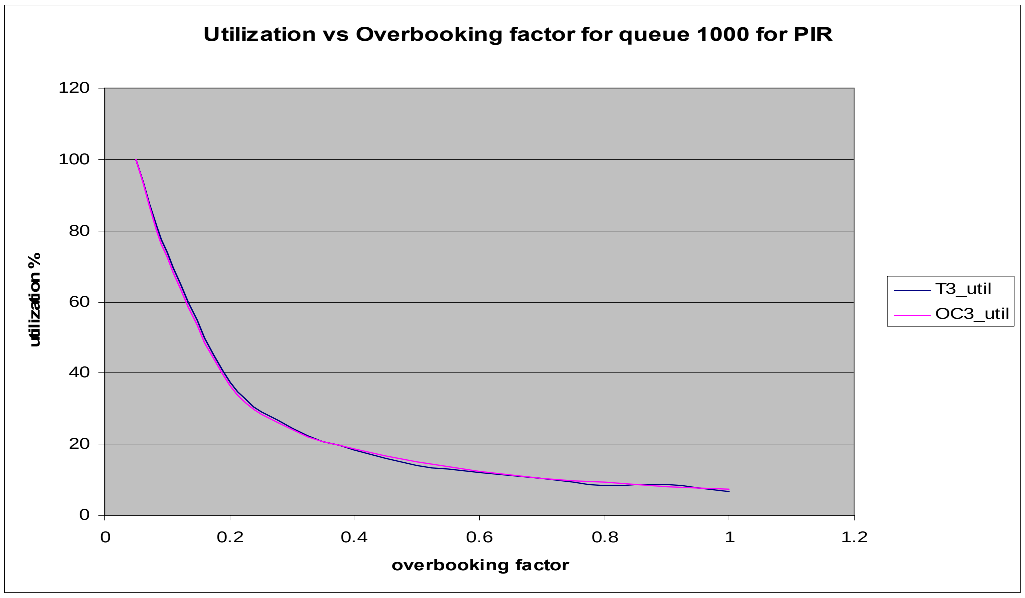
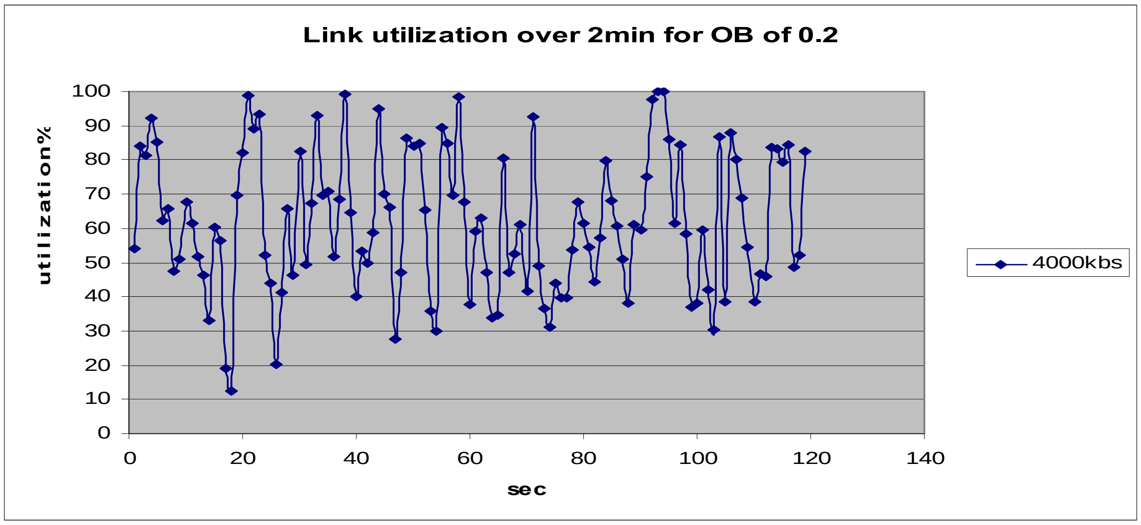


Figure 4.2 Average utilization for OC3 and T3 links. Both links have a similar average utilization with the same overbooking factors.

Although different trunks have similar average utilization shown in Figure 4.2, their instantaneous utilization is significantly different. Figure 4.3 shows instantaneous utilization for the two trunks. They were configured with exponential sources with an overbooking factor of 0.2. Both trunks have similar average utilization around 64%. The larger link with 400Mbps has more stable instantaneous utilization between 58% and 68% whereas the smaller link with 4Mbps has higher variance with instantaneous utilization between 13% and 99%.



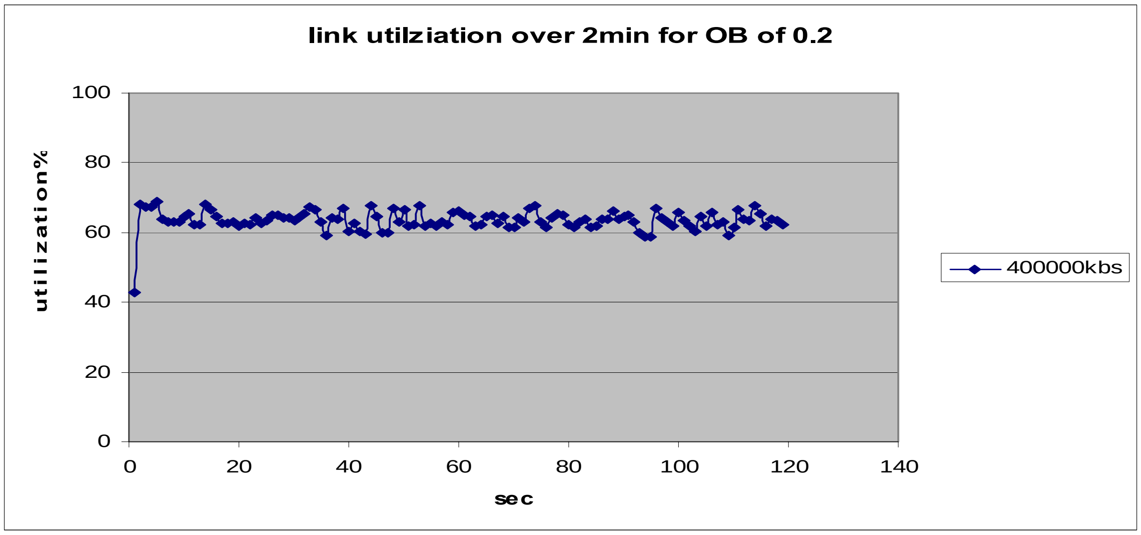
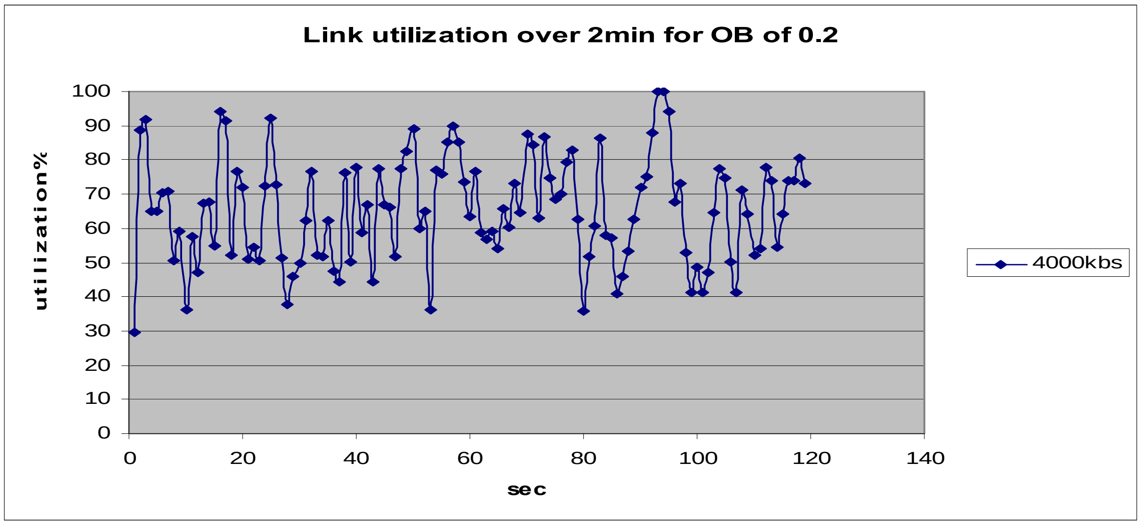


Figure 4.3: Instantaneous utilization for two links with an overbooking factor of 0.2 for exponential sources

Figure 4.4 shows results for self-similar sources with a Pareto parameter of 1.5. The self-similar traffic also shows different instantaneous utilization for different trunk sizes. Both trunks have similar average utilization around 65%. Utilization for the 400Mbps link varies between 59% and 71% whereas the 4Mbps link swings between 36% and 99%.



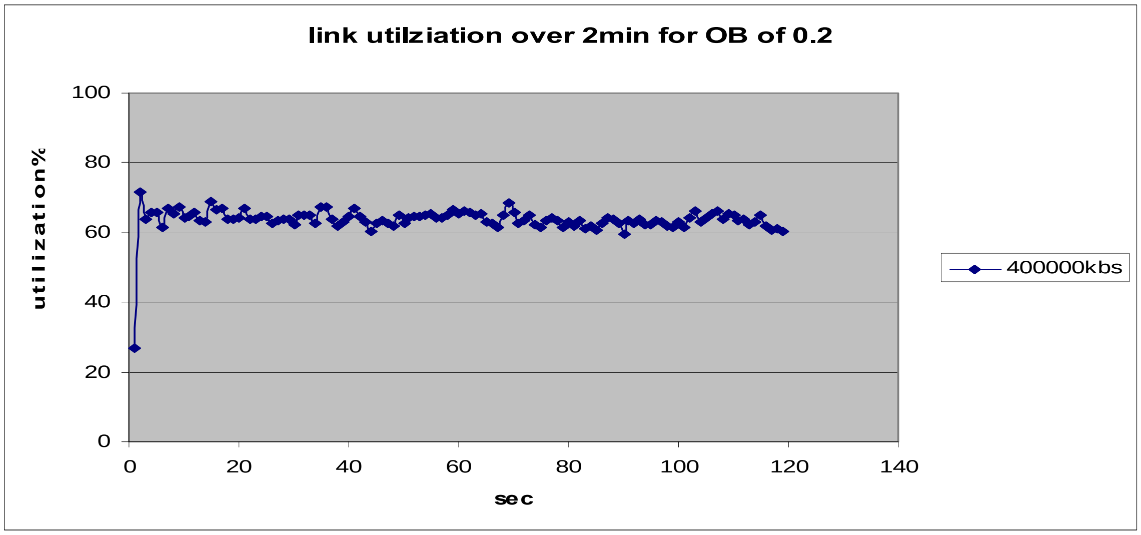


Figure 4.4: Instantaneous utilization for two links with an overbooking factor of 0.2 for self-similar sources

## 4.5 Summary

By definition, the effective bandwidth of sources decreases when the link capacity increases. This suggests that larger overbooking factors can be used for large trunks. The simulation results show that large trunks have lower packet loss probabilities than small trunks when they have similar average utilization.

# Chapter 5 Piecewise Linear Simulation

In this chapter, simulation experiments with piecewise linear overbooking are examined. Piecewise linear overbooking results in superior performance when compared with a single overbooking factor.

## 5.1 Overbooking Algorithm

When admitting a flow, a random source was chosen out of the trace file with 400,000 virtual circuits represented. When a new connection is admitted into backbone, its overbooked bandwidth was subtracted from available trunk bandwidth. This process continues until the available bandwidth reaches zero or some small value.

### 5.1.1 Single Value Overbooking

Currently, network providers use a single overbooking value for admission control. The same overbooking value is applied to all circuit connections. This approach is simple and fast. However, this approach does not take into consideration the usage pattern for different users. As a result, the same overbooking factor may lead to inferior network performance for different networks at different times. When most admitted connections are large CIR circuits, the link will be lightly loaded. When most connections are small CIR circuits, the link may get congested. Both low utilization and poor QoS may lead to lost avenue for network providers.

### 5.1.2 Piecewise Overbooking

The statistical analysis of network traffic suggests that different overbooking values should be used for VCs with different CIR values. When usage patterns are taken into consideration, better utilization for trunks can occur.

Routers can be configured to store information on usage patterns. The profile will contain the average utilization of virtual circuits for each class of CIR and the corresponding overbooking factor. The profile is dynamically updated after some period of time. The original values may be obtained from reference values from the hardware vendor. In real systems, the class of CIR is limited and each type of CIR can be assigned a value based on its utilization. However, when the classes of CIR are large in number, a group of similar CIRs may be assigned to the same value to reduce complexity. A sample usage profile is presented in Table 5.1.

| CIR(Kpbs) | OB |
|---|---|
| 4 | 2.3 |
| 128 | 0.33 |
| 256 | 0.28 |
| 1544 | 0.1 |

Table 5.1. A sample usage profile table

As indicated in Chapter 3, network administrators may have a target performance objective. A utilization of 80% was chosen as the maximum trunk utilization. This value combined with other parameters such as circuit utilization will determine the overbooking factor for each kind of CIR circuits.

## 5.2 Experiment Results

Simulation experiments for the piecewise overbooking and single value overbooking were conducted using sample data and trace data. The results show significant performance differences between the two schemes.

The experimental results in Figure 5.1 show samples taken from two pools representing two utilization regions for test data.  The utilization from the two-factor piecewise overbooking scheme has lower variance compared to two single value overbooking schemes.  For this experiment, the 40% overbooking value resulted in low utilization under 80% with the average of only 50%. For the more aggressive 20% overbooking value, the 80% utilization objective was violated 60% of the time. Significant packet loss occurs when the utilization approaches 100%.

The piecewise linear approach increases average utilization when compared to the 40% overbooking value while avoiding peak values from the 20% overbooking factor that causes poor quality of service. The utilization values from the two-factor (40% + 15%) nonlinear overbooking remain very stable around 70%..

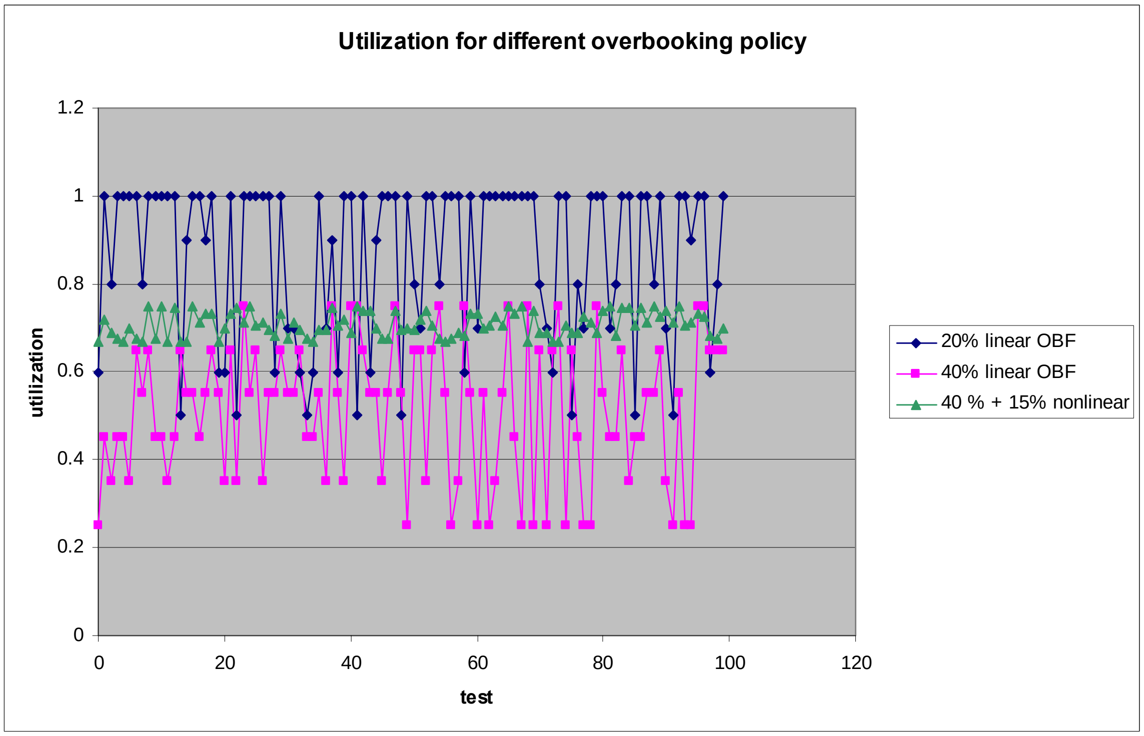


Figure 5.1: Performance compassion for piecewise and single overbooking.

Figures 5.2a&b are simulation results using backbone trace data. Figure 5.2a uses piecewise linear overbooking values during admission. Both experiments run the dumbbell simulation 100 times. For the piecewise linear overbooking approach, utilization performance is rather stable. The average utilization is 72.68% with standard deviation and variance at 0.020 and 0.000648 respectively. Figure 5.2b represents the results from the single value overbooking. This policy results in higher variance for similar average utilization. The average utilization is 71.18% with standard deviation and variance at 0.116 and 0.023 respectively.

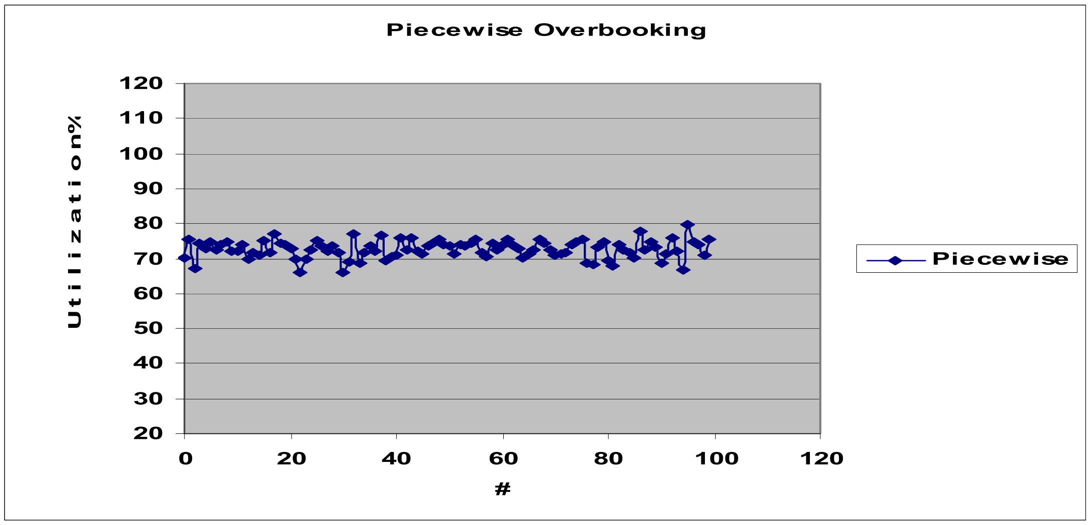


Figure 5.2a: Performance results for the piecewise linear overbooking from trace data

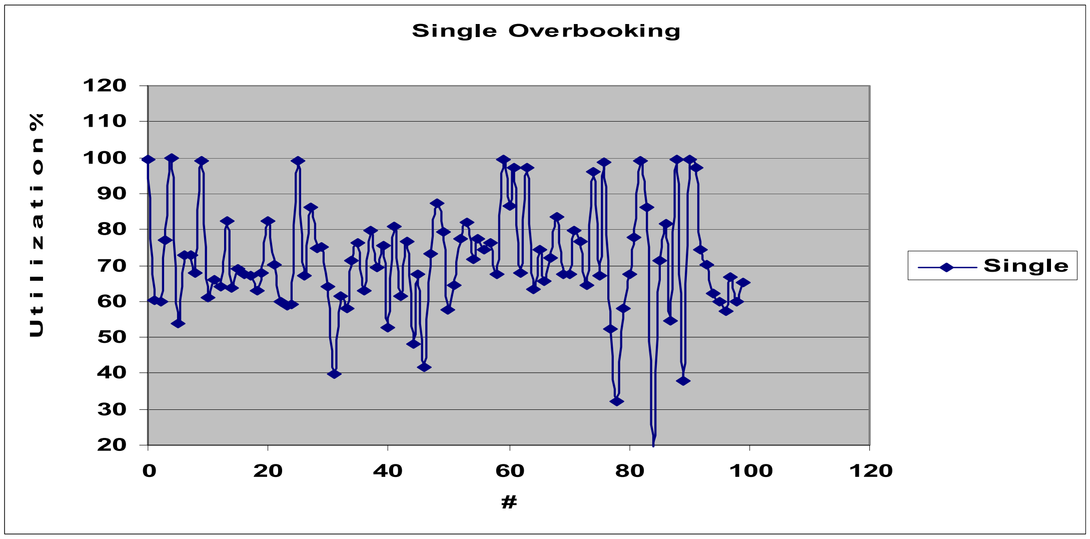


Figure 5.2b: Performance results for the single value overbooking from trace data

## 5.3 Summary

Simulations in this chapter detail the impact of different overbooking policies on network performance. Piecewise linear overbooking results in better network performance than single value overbooking.

# Chapter 6 Conclusions and Future Work

## 6.1 Conclusions

Overbooking can significantly increase bandwidth utilization and profits for network providers. This research proposes the new overbooking policy for network providers to use.  Piecewise linear overbooking factors for different CIR circuits results in higher utilization while maintaining similar quality of service. This new approach can help network providers predict network performance more accurately.

This research uses effective bandwidth analysis to show that large trunks can tolerate higher overbooking factors. Existing network hardware options should be used to select different overbooking factors based on trunk size.

## 6.1 Future Work

Statistical analysis and simulation experiments show that new overbooking policies can benefit network providers. Future work will implement these policies in real networks. Although this research provides guidelines, many other factors must be considered in real implementations. Those factors include usage profile design, the size of networks and real time traffic models. For example, corporation virtual circuits send traffic during daytime hours whereas home links have higher usage during the night time and weekends. When overbooking these two kinds of traffic, network providers may need to take time of day into consideration. Finally, a better approach is needed to predict the utilization for new accepted circuits. In this research, these values were retrieved from previous trace data. In real systems, trace data size may be prohibitive.

## Bibliography

[CIS04] Cisco Systems, *"Traffic and resource management", January 2, 2004 ,* <http://www.cisco.com/en/US/products/hw/switches/ps718/products_implementation_design_guide09186a00800ad9be.html>

[CIS05] Cisco Systems, "Cisco line cards data sheets", *December 20, 2005, <http://www.cisco.com/en/US/products/hw/modules/ps2710/products_data_sheets_list.html>*

[CRO97] Mark E. Crovella and Azer Bestavros, *"Self-similarity in World Wide Web traffic: Evidence and possible causes," IEEE/ACM Transactions on networking*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[CS99] C. Courcoubetis and V.A. Siris. *``Measurement and analysis of real network traffic''. In Proc. of 7th Hellenic Conference on Informatics (HCI'99), Ioannina, Greece, August 1999. Available as ICS-FORTH TR-252*

[NS06] "The Network Simulator ns-2", January 30, 2006, <http://www.isi.edu/nsnam/ns/index.html>

[FLO01] Sally Floyd and Vern Paxson, "Difficulties in simulating the Internet". IEEE/ACM Transactions on Networking, 9(4), Aug. 2001.

[GIE78] Giessler, A., Haanle, J., Konig, A., and Pade, E. "Free buffer allocation – An inversigation by simulation," *Compu. Networks 1, 3 (July), 191 - 204*

[JAI88] R. Jain and K. Ramakrishnan, *"Congestion Avoidance in Computer Networks with A Connectionless Network Layer: Concepts, Goals, and Methodology," Proc. Computer Networking Symposium, Washington, D.C., April 11-13, 1988, pp. 134-143,*

[KLE79] Kleinrock, L. "Power and deterministic rules of thumb for probabilistic problems in compute communications," *In proceedings of the Internatinal Conference on Communications (June 1979), pp.43.1.1- 43.1.10*

[LEE99] Sandeep Bajaj, Lee Breslau, etc, *"Improving Simulation for Network Research*". USC Computer Science Department Technical Report, 1999.

[LEL94] W.Leland, M. Taqqu, W, Willinger, and D. Wilson, *"On the self-Similar Nature of Ethernet Traffic".* IEEE/ACM Transactions on Networking, vol. 2, pp 1-15, 1994

[LIT61] Little, D. C. "A proof of the Queueing Formula: L = W", *Operations*

*Research, 9 (1961): 383 – 387*

[MIS96] Mischa Schwartz, *"Broadband Integrated Networks."* Prentice Hall PRT, New Jersey, 1996.

[NOR94] I. Norros, *"On the Use of Fractional Brownian Motion in the Theory of Connectionless Networks." IEEE Journal on Selected Areas in Communications, August, 1995.*

[ODL00] Odlyzko, A. M. "*The Internet and other networks: Utilization rates and their implications", Information Economics & Policy 12 (2000), 341-365.*

[PAX95] Vern Paxson and Sally Floyd, "*Wide-Area Traffic: The failure of Poisson Modeling"*, IEEE/ACM Transactions on Networking, vol. 3 pp 226-244, 1995.

[ROM94] Romanow, A., and Floyd, S., *"Dynamics of TCP Traffic over ATM Networks."* IEEE JSAC, V. 13 N. 4, May 1995, p. 633-641.

[SAL03] saltlakejohn, June, 2003, *"Overbooking bandwidth, Ethics or Marketing Question?",* September 30, 2003, *<http://www.webhostingtalk.com/showthread.php?threadid=168761>*

[SIR00] Vasilios A. Siris, 2000 "*Large Deviation Techniques for Traffic Engineering*". December 26, 2005, <http://www.ics.forth.gr/netlab/msa/index.html>

[WIL98] William Stallings, *"High-speed Networks, TCP/IP and ATM Design Principles."* Prentice Hall PRT, New Jersey, 1998.

## Appendix A

1. *ns* script for piecewise overbooking

```
ns-random 0
set ns [new Simulator]

set simtime             [lindex $argv 0]
set sampleinterval      [lindex $argv 1]        ;# SampleInterval: 0.1-
1sec, but xg name uses snapelInterval*10 refer old code from Dai

set queuelimit          [lindex $argv 2]        ;# the queue for vc and
trunks will have same size.
set bandwidth           [lindex $argv 3]        ;# bandwidth of
bottleneck link

set udppercent          [lindex $argv 4]        ;# % of udp in the src
of srcNum. (0.10 = 10% of srcNum is udp)
set tcpflavor           [lindex $argv 5]        ;# 1: Tahoe      2:
Reno  3: Vegas
set traffictype         [lindex $argv 6]        ;# 1: exponential 2:
pareto types      3: CBR
set packetsize                [lindex $argv 7]        ;# 800 in bytes
seems suitable. 1000 is maximal for UDP or 64K?

set filename            [lindex $argv 8]        ;# Input data file
set aggfactor           [lindex $argv 9]        ;# Aggregate mutiple
flows into one to decrease # of nodes
set obfactor            [lindex $argv 10]       ;# 0.17 means 17%
set doInstance                [lindex $argv 11]

# initialize the data in input file into array and initialize the
traffic sources
proc initialize { ns } {
      global bandwidth
      global traffictype packetsize
      global filename aggfactor
      global obfactor


      global srcnum                       ;# number of source (combined
flows(vcs)) == source node numbers
      global totalvcnum             ;# total number of vcs accepted,
not the sources setup: totalvcnum / 15 ~~ srcnum
      global app_                         ;# application lists

      #initialization
      set srcnum 0
      set totalvcnum 0

      set avgsum 0                  ;# the average bandwidth
for each source
      set pirsum 0                  ;# the PIR bandwidth (sum of VCs)
for each source
```

```
      set rownum 0                  ;# the number(index) of rows of
data in input file
      set aggvccount 0        ;# aggrefactor or less. flow(VC) count is
the number for aggregated CIRs into one source: aggvccount <= aggfactor

      #setup CIR profiles
      # This suppose to be target utilization like 80%(0.8), but need
to be modified because of inaccuracy of trace data.
      set tutil 1.8     ;#2.0 ;#1.6


      set pfCIR_(0) 4;        set pfOB_(0) [expr 4.58 / $tutil]
      set pfCIR_(1) 16;       set pfOB_(1) [expr 1.91 / $tutil]
      set pfCIR_(2) 32;       set pfOB_(2) [expr 0.52 / $tutil]
      set pfCIR_(3) 64;       set pfOB_(3) [expr 0.52 / $tutil]
      set pfCIR_(4) 128;            set pfOB_(4) [expr 0.37 / $tutil]
      set pfCIR_(5) 256;            set pfOB_(5) [expr 0.36 / $tutil]
      set pfCIR_(6) 384;            set pfOB_(6) [expr 0.30 / $tutil]
      set pfCIR_(7) 512;            set pfOB_(7) [expr 0.32 / $tutil]
      set pfCIR_(8) 768;            set pfOB_(8) [expr 0.26 / $tutil]
      set pfCIR_(9) 1024;           set pfOB_(9) [expr 0.21 / $tutil]
      set pfCIR_(10) 1536;    set pfOB_(10) [expr 0.14 / $tutil]

#     set randCIRIndex [expr int([expr 11 * [expr rand()]])]
#     set randCIR $pfCIR_($randCIRIndex)

      set bw [expr $bandwidth * 1.000]          ;# bandwidth for
calculation.
      set bwlimit 10                                  ;# smallest size
of flow that can be accepted
      set bwlimit [expr $bwlimit * $obfactor] ;# smallest size of flow
that can be accepted after overbooking
      # assigne the cir and avg from input file
      # for smalldata.txt(cir_avg.dat),   0: cir 1: pir     2: avg load
3: avg load
      # for largedata.txt(outdb.txt),     0: cir 1: pir     2: avg load
3: weighted avg load    :: This is data calculated by Casey
      # for largedataxxxx.txt,                  0: cir 1: pir     2:
avg load 3: avg load weighted by x.xxx
      set fid     [open $filename r]
      while { [eof $fid] == 0 } {
            gets $fid rowstring
            set vals [split $rowstring \t]
            set cir_($rownum) [lindex $vals 0]
            set pir_($rownum) [lindex $vals 1]
            set lod_($rownum) [lindex $vals 2]  ;# average load
            set avg_($rownum) [lindex $vals 3]  ;# weighted average
load
            incr rownum
      }
      close $fid



      while { $bw > $bwlimit } {    ;# if it is 4 or less, we do not
need to add it to avoid too aggresive
```

```
            # randomly select the data array index

            set i [expr int([expr [expr $rownum - 1] * [expr rand()]])]
      #     while {$cir_($i) != $randCIR} {
      #           set i [expr int([expr [expr $rownum - 1] * [expr
rand()]])]
      #     }

##          puts "$randCIR $cir_($i) $bw"

            set pfi 0
            while { $cir_($i) > $pfCIR_($pfi) && $pfi < 10 } {
                  incr pfi
            }

            # set obfactor for each kinds of CIR flows
            set obfactor $pfOB_($pfi)

            set bookedbw [expr $obfactor * $cir_($i)]


            # accept the acceptable flows with
            # smaller cir:    potential flow should have smaller CIR
than available bandwidth
            # nonzero cir:    guanrantee the each flow is used only
once. Cir will be set to zero after selection.
            # nonnega avg:  valid traffic. Avoid use invalid data like
-1
            if { $bw >= $bookedbw && $cir_($i) > 0 && $avg_($i) >= 0 }
{
                  set bw [expr $bw - $bookedbw]
                  set avgsum [expr $avgsum + $avg_($i)]
                  set pirsum [expr $pirsum + $pir_($i)]
                  incr totalvcnum

                  if { $avg_($i) > 0 } {
                        incr aggvccount
                  }

                  # set up source when nonzerovccount reach aggfactor
or remaining bandwidth reach its limit
                  if { $aggvccount == $aggfactor || $bw <= $bwlimit } {

                        # on/off two state traffic: expenonetial and
pareto
                        if { $traffictype == 1 || $traffictype == 2 } {

                              set baserate $pirsum
                              set burstms       [expr int([expr 1000 *
[expr $avgsum * 1.000/ $baserate]])]

                              if { $burstms > 1000 } {
##                                  puts "burstms:$burstms should not
be greater than 1000"
                              }
                              if { $baserate > $bandwidth } {
                                    #puts "baserate: $baserate shouldnt
be larger than bandwidth: $bandwidth flowcount: $aggvccount avgsum:
$avgsum pirsum: $pirsum srcnum: $srcnum"
                              }
                              # puts "avgsum: $avgsum, burstms:
$burstms, baserate(pirsum): $baserate"

                              if { $traffictype == 1 } {
                                    set app_($srcnum) [new
Application/Traffic/Exponential]
                              }
                              if { $traffictype == 2 } {
                                    set app_($srcnum) [new
Application/Traffic/Pareto]
                                    $app_($srcnum) set shape_ 1.5
                              ;#    set app_($srcnum) shape_ 1.5
                              }

                              $app_($srcnum)    set packetSize_
$packetsize
                              $app_($srcnum)    set burst_time_ [expr
$burstms]ms
                              $app_($srcnum)    set idle_time_    [expr
1000 - $burstms]ms
                              $app_($srcnum)    set rate_ [expr
$baserate]Kbs
                        #     if { $traffictype == 1 } {
                        #           $app_($srcnum)    set burst_time_ 0
                        #           $app_($srcnum)    set rate_ [expr
[expr $baserate * 1000000]]Kbs
                        #     }

                        } ;# end of if traffictype 1 or traffictype 2

                        if { $traffictype == 3 } {

##                            puts "CBR traffic types"
                              set app_($srcnum) [new
Application/Traffic/CBR]
                              $app_($srcnum)    set packetSize_
$packetsize
                              $app_($srcnum)  set rate_ [expr
$avgsum]Kbs
                              if { $avgsum > $bandwidth } {
##                                  puts "avgsum: $avg shouldnot be
larger than bandwidth: $bandwidth"
                              }
                        }

                        if { $traffictype == 4 } {

##                            puts "FTP traffic types"
                              set app_($srcnum) [new Application/FTP]
                        }
                        #puts "bw: $bw, srcnum: $srcnum, baserate:
```

```
$baserate, pirsum: $pirsum, avgsum: $avgsum, aggvvcount: $aggvccount"
                        ;#one flow is setup, set and reset the
variables
                        incr srcnum
                        set avgsum  0
                        set pirsum  0                ;# for baserate
                        set aggvccount    0          ;# 15 total
                  } ;# end of if aggrated flow to source
                  set cir_($i) 0          ;# To avoid the redundantly
selecting same source again and again
            } ;# end of if accept a flow
      } ;# end of while compute and accept flow
##    puts "Initialize done with $srcnum Sources and $totalvcnum flows"
}
initialize $ns

proc build_topology { ns } {
      global queuelimit bandwidth
      global srcnum

      global node_

      # setup routers
      set node_(r1) [$ns node]
      set node_(r2) [$ns node]
      $ns duplex-link $node_(r1) $node_(r2) [expr $bandwidth]Kb 30ms
DropTail    ;# check the queue here.
      $ns queue-limit $node_(r1) $node_(r2) $queuelimit
      $ns queue-limit $node_(r2) $node_(r1) $queuelimit

      for { set sourindex 0 } { $sourindex < $srcnum } { incr
sourindex } {
            # why maximal 155250Kb here for bandwidth
            set node_($sourindex)   [$ns node]
      #     $ns duplex-link $node_(r1) $node_($sourindex) [expr
$bandwidth]Kb 10ms DropTail         ;# srcNum is 15
            $ns duplex-link $node_(r1) $node_($sourindex) 155250Kb 10ms
DropTail                      ;# 00 ------ 15
            $ns queue-limit $node_(r1) $node_($sourindex) 30      ;#
this is useless                     ;# 01 ------ 16
            $ns queue-limit $node_($sourindex) $node_(r1) 30      ;#
this is useless                     ;# 02 ------ 17

            set destindex     [expr $sourindex + $srcnum]
            set node_($destindex) [$ns node]
      #     $ns duplex-link $node_(r2) $node_($destindex) [expr
$bandwidth]Kb 2ms DropTail          ;# 14 ------ 29
            $ns duplex-link $node_(r2) $node_($destindex) 155250Kb 2ms
DropTail
            $ns queue-limit $node_(r2) $node_($destindex) 30      ;#
this is useless
            $ns queue-limit $node_($destindex) $node_(r2) 30      ;#
this is useless
      }
##    puts "build_topolody done"
}
```

```
build_topology $ns

# monitor the ingress queue from r1 to r2. I see other use "" instead
of 0
set qmon [$ns monitor-queue $node_(r1) $node_(r2) 0]

proc create-source { ns stime } {
      global simtime
      global udppercent tcpflavor packetsize
      global bandwidth

      global srcnum
      global node_ app_

      global tcp_ sink_ udp_ null_

      # set packetsize 1050         ;# this size is same as the size
for traffic. Hence no divide and assemble between transport layer and
applcation layer.
      set windowSize [expr $bandwidth /100]     ;# T3:      447
      if { $windowSize < 3 } {                        ;# OC3:     1552
#           set windowsSize 3
#     }

      # UDP staff
      set udpSrcNum [expr int([expr $udppercent * $srcnum])]
      for {set i 0} {$i < $udpSrcNum} {incr i} {
            set udp_($i) [new Agent/UDP]
            $ns attach-agent $node_($i) $udp_($i)
            set null_($i) [new Agent/Null]
            set dest [expr $i + $srcnum]        ;# equation and number
are exactly same with build_topology. $i + $srcnum
            $ns attach-agent $node_($dest) $null_($i)
            $ns connect $udp_($i) $null_($i)    ;# transport layer need
to have sender/receiver, hence it can transport application traffic

            $app_($i) attach-agent $udp_($i)    ;# traffic is
application, it send to transport agent like TCP/UDP
            $ns at $stime "$app_($i) start"
      }
      #TCP stuff
      for {set i $udpSrcNum} {$i < $srcnum} {incr i} {
            if { $tcpflavor == 1 } {
                  set tcp_($i) [new Agent/TCP]
            }
            if { $tcpflavor == 2 } {
                  set tcp_($i) [new Agent/TCP/Reno]
            }
            $tcp_($i) set window_ 10      ;#$windowSize     max bound
on window size the default is 20
      #     $tcp_($i) set packetSize_ $packetsize     ;# default 1000
            $ns attach-agent $node_($i) $tcp_($i)
            set tcpsink_($i) [new Agent/TCPSink]      ;# they have ACK
packetSize_
            set dest [expr $i + $srcnum]
            $ns attach-agent $node_($dest) $tcpsink_($i)
            $ns connect $tcp_($i) $tcpsink_($i) ;# create a
link/circuit here

            $app_($i) attach-agent $tcp_($i)
            $ns at $stime "$app_($i) start"
      }
##    puts "create_source done"
}

# Some global vars:
set starttime     0.0
set meastime      $sampleinterval         ;# we need this to remember
last time and measure the results

set totaldeparturekilobytes   0     ;# to avoid out of bound limit
set maxinstantutil 0                      ;# for instantaneous
variables
set mininstantutil 100

# recursive call is also to avoid out of bound.
proc start {} {
      global simtime sampleinterval
      global bandwidth
      global doInstance

      global srcnum totalvcnum

      global ns qmon
      global meastime


      global instantfile
      global maxminfile

      global totaldeparturekilobytes
      global maxinstantutil mininstantutil
      global maxinstanttime mininstanttime


      $qmon instvar bdepartures_ bdrops_ barrivals_ pdepartures_
pdrops_ parrivals_

      set now [$ns now]

      # we decrease the number by 1000 here otherwise the total number
will run out of bound
      set totaldeparturekilobytes [expr $bdepartures_ * 0.001 +
$totaldeparturekilobytes]
      set instantdeparturekilobytes [expr $bdepartures_ * 0.001]

      set tmp [expr $now + $sampleinterval]
      if { $tmp >= $simtime } {     ;# recursive call until the final
time is reached, print the output after time is reached
            set util [expr $totaldeparturekilobytes / $now / $bandwidth
* 8 * 100]
#           set util [expr $pdepartures_ / $now / $bandwidth / 1000.0 *
8040 * 100]
            set dropPct [expr 1.000 * $pdrops_ / $parrivals_ * 100.0]

            if { $parrivals_ < 0 } {
##                puts "parrivals is negative $parrivals_"
            }
#           puts [format "%5.2f %5.2f %4.0f %5.0f" $util $dropPct
$srcnum $totalflownum]
            puts "$util"      ;# $dropPct $srcnum"
      }

      if { $doInstance }      {
            set instantutil         [expr $instantdeparturekilobytes /
$sampleinterval / $bandwidth * 8 * 100]
            puts $instantfile [format "%.2f  %.4f" $now $instantutil]

            if { $instantutil > $maxinstantutil} {
                  set maxinstantutil [format "%.4f" $instantutil]
                  set maxinstanttime [format "%.2f" $now]
            }

            if { $instantutil < $mininstantutil && $now > 1} {

                  set mininstantutil [format "%.4f" $instantutil]
                  set mininstanttime [format "%.2f" $now]
            }
      }

      $qmon set bdepartures_ 0
      $qmon set pdepartures_ 0

      set meastime [expr $meastime + $sampleinterval]
      $ns at $meastime "start"                                    ;#
recursive calls
}


# setup out instantaneous data

if { $doInstance } {

##    puts "The experiments for trace file"
      set tracefilename
      "../ns_result/queue${queuelimit}/instant_${bandwidth}kbs_[format
%.1f $sampleinterval]ps_[format %.2f
$udppercent]udp_${traffictype}tt_[format %.2f $obfactor]obf.tr"
      #set maxminfilename           "instant_${bandwidth}kbs_[format
%.2f $obfactor]obf.mm"

      set instantfile   [open $tracefilename w]
      #set maxminfile   [open $maxminfilename w]

##    puts $instantfile "\"${bandwidth}kbs_${obfactor}obf\""
#     puts $maxminfile "\"${bandwidth}kbs_${obfactor}obf\""
#}
```

```
proc finish {} {
      global ns

      global instantfile maxminfile
      global tracefilename maxminfilename

      global maxinstanttime maxinstantutil
      global mininstanttime mininstantutil

      global doInstance

if { $doInstance } {
      #     puts $maxminfile "$maxinstanttime $maxinstantutil"
      #     puts $maxminfile "$mininstanttime $mininstantutil"
#     puts $instantfile "\"${bandwidth}kbs_${obfactor}obf\""
#     puts $maxminfile "\"${bandwidth}kbs_${obfactor}obf\""
##          puts $instantfile "\n";
##          puts $instantfile "\"min: ${mininstanttime}s
$mininstantutil\""
##          puts $instantfile "$mininstanttime $mininstantutil"

##          puts $instantfile "\n";
##          puts $instantfile "\"max: ${maxinstanttime}s
$maxinstantutil\""
##          puts $instantfile "$maxinstanttime $maxinstantutil"

            close $instantfile
      #     close $maxminfile

            exec xgraph $tracefilename &
      #     exec xgraph "../ns_result/test.txt" &
#     }

      # $ns flush-trace
      # close $nf
      # exec nam out.nam &

      exit 0
}

# initialize the qmon varialbe, seem unnecessary
$ns at $starttime       "$qmon set bdepartures_ 0;    $qmon set
pdepartures_ 0"
$ns at $starttime "$qmon set bdrops_ 0;         $qmon set pdrops_ 0"
$ns at $starttime       "$qmon set barrivals_ 0;      $qmon set
parrivals_ 0"

$ns at $starttime       "create-source $ns $starttime"      ;# start
traffic
$ns at $meastime  "start"                                         ;#
start measurement
$ns at $simtime   "finish"                                  ;# collect
reslut data
$ns "run"
```