# QUALITY OF SERVICE TESTING METHODOLOGY

BY

Ankur Chadda

B.E., University of Bombay(Mumbai), India (1999)

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science

in

Computer Science

December, 2004

This thesis has been examined and approved.

Thesis director, Dr. Radim Bartos
Associate Professor of Computer Science

Dr. Robert Russell
Associate Professor of Computer Science

Mr. Benjamin Schultz
Consortium Manager, InterOperability Laboratory,
Research Computing Center

Date

# Dedication

To my family Vinod, Madhu, Anshul Chadda and Dorothée.

# Acknowledgments

I would like to express my sincere gratitude to Dr. Radim Bartos without whose guidance this thesis would not have been possible. It has been a pleasure to work with him. I would also like to thank Dr. Robert Russell and Mr. Benjamin Schultz for their guidance. I am also grateful to the InterOperability Laboratory for providing me the required resources for the research and supporting me for the past three years. Finally, I would like to thank Peter Scruton, Chris Volpe, Paul Scarff and Neil Carter for the numerous helpful discussions.

## TABLE OF CONTENTS

# List of Tables

# List of Figures

# ABSTRACT

## QUALITY OF SERVICE TESTING METHODOLOGY

by

Ankur Chadda
University of New Hampshire, December, 2004

Typically, a network service provider offers best-effort service to customers and Quality of Service (QoS) is provided only by network over provisioning. When traffic bursts cause congestion, this default approach does not always provide an acceptable solution. As the trend in networking is convergence of voice, video and data, it is important to differentiate traffic into high and low priority classes through QoS mechanisms. To deliver differentiated QoS, the routers utilize classification and sometimes remarking of the traffic. Based on service agreements with the customer, a certain class of traffic is then given more resources by the router. Evaluation of the results of these methods and approaches on the network performance is very important to device designers, network evaluators, network administrators, network operators and others involved.

To study the effects of QoS solutions/techniques on the traffic, methods for testing need to be developed. The objective of this thesis is to propose methods of measuring QoS metrics like throughput, latency, and jitter for different types of traffic through a device. Classification and remarking were observed to put extra load on a device. For some devices, the performance deteriorates when classification and remarking are enabled. Throughput was seen to drop for classification and remarking in some cases. In multiple instances, latency and jitter increased for classification and remarking. There are multiple ways of measuring jitter which were also discussed in the methodology development. It was also observed that some definitions which might seem better from a research perspective are not always available to measure or widely used in the industry. Based on the study it was

concluded that service/network providers will have to take care while providing QoS to the customers in order avoid the scenarios discussed here.

# Chapter 1

# Introduction

An assumption has developed over the last decade that the Internet can support any form of communication service. The services range from traditional reliable, non-real time data transfer from one computer to the other, to time critical services such as audio and video streaming. These services require the network to negotiate certain performance attributes between end applications and the core. Quality of Service (QoS) as the name suggests aims to provide performance attributes in the form of service guarantees that providers can offer to their customers. Network routers have had QoS implementations for some time now, but there has been no universally agreed upon metric or methodology with which to test QoS. This thesis proposes a testing methodology for gathering and analyzing relevant QoS data and, if possible, suggesting changes to improve performance.

## 1.1 Quality of Service

Quality of Service can be thought of as a mechanism to satisfy a level of service performance across a network. It can also be understood as the ability of a network to provide desired handling of the traffic such that the network meets the level of service that end applications expect.

## 1.2 Early days

In the early iterations of the Internet protocol (IP), getting a packet to its destination was of primary concern. Internet Service Providers (ISP) were more involved in keeping up with the increasing demands of the customers and keeping the network up and running.

IP's initial design included a Type of Service (TOS) field intended for offering different categories of treatment. Using this feature, however, was not a priority for service providers. At the same time, over-engineering hampered the deployment of QoS; a network was always provisioned for more traffic than it actually had to handle to avoid congestion and contention for resources. As contention of resources is one of main reason for QoS to be deployed, it was avoided by over engineering.

Another, relatively minor factor in the slow growth of QoS was the complexity of billing systems. Originally, providers had fairly simple billing systems, based on just time and access capacity. Migrating to a billing system smart enough to differentiate between varied service plans and customers was seen as prohibitively difficult, given that it might require the system to look into 'almost' packet level granularity.

## 1.3 How to deliver QoS?

A QoS-enabled network must be able to handle different traffic streams in different ways. This necessitates categorizing traffic into types or classes and defining how each type is handled. All the following aspects could be considered within the scope of QoS [1]:

- Differentiation of traffic (Classification)
- Admission control
- Queuing

- Congestion management

The following sections explain these aspects in further detail.

### 1.3.1 Differentiation of Traffic

IP routers are designed to treat all packets in the same way. Packets arriving at the router are considered as new event without having any local state memory. So the basic building block here is the ability to classify the traffic into different categories based on what service they are to receive. Once the packets have been classified they can be given differential treatment based on the classification.

One of the easiest ways to classify a packet is to embed the packet header with an information field that specifies the class of traffic the packet belongs to. This way of classification is known as packet service classification. The advantage of this method is that the packet need not be classified at any router in the network; instead appropriate treatment can be given to the packet based on the service classification field. Differentiated Services architecture [6] could be considered as the IETF's initiative towards standardizing this process.

This method can be problematic if an excessive number of applications are sending packets expecting premium handling. Taking in packets and assigning them specific behavior based on a value set in the packet header can tax the router's resources. Another stumbling block is that different networks may use different codes to signify the same category of service. In this case, the TOS or differentiated service field might be remarked before being admitted to a network. This marking or remarking is often involved in controlling congestion.

The classification of a packet into a service class can be done based on the following parameters [1]:

- Service Mark

- Protocol
- Destination protocol port
- Source protocol port
- Destination host address
- Source host address
- Source device interface
- Any combination of above

### 1.3.2 Admission Control

When a packet arrives at the ingress of a network, it can be classified into two categories, namely in profile and out of profile. If a packet conforms to an agreement with the owner (service provider) of the network while entering, it will be classified as in profile and will be allowed to enter the network with or without any changes to the packet. For example, the ingress router can be configured to allow only TCP packets to go through. In this case any TCP packet arriving at the ingress router is classified as in profile and all other are considered out of profile. But if the packet does not conform then it can be dropped at the entry point into the network. The ingress router can be configured to utilize any of the parameters listed above for classification to determine whether the packet is in or out of profile.

Sometimes packets may be dropped based on the scarcity of the resources available to handle them, regardless of whether a packet is in profile or out. There are a few ways in which a device can judge the available resources of the network.

### 1.3.3 Queuing

Every router is a shared resource in a network and any packet reaching a given router can be considered as request for service. If the route engine or processor is busy servicing other requests, the arriving packet must wait until the processor is ready to service the request. Such requests wait in a queue until then. If the rate at which all the packets are arriving at the router is greater than the rate at which they are being processed, the queue will start filling up. Once the queue is full, the packets will start getting discarded.

To help assuage this, the router can use a queuing discipline to handle requests. A queuing discipline can be designed to give preference to one category of traffic and not to another. Managing the queuing system is one of the fundamental aspects that determine the quality of provided service as it creates the basic differentiation between the service levels of different classes of traffic. There are quite a few ways of queuing. We discuss some important ones below.

#### First In First Out (FIFO) queuing

This is the most commonly followed queuing discipline. Very large numbers of network devices implement this type of queuing. Here the order in which the packet arrives is maintained and there is no preference given to any packet. This means there is no differential treatment given to any packet. This form of queuing is also known as first come first served (FCFS) queuing. There are some issues with this kind of queuing as there is no differentiation between the packets. For example, TCP may back off because of congestion while UDP packets still continue to flood the network, in turn using up the entire bandwidth. TCP is a protocol which tries to adjust based on the amount of traffic in the network, while UDP does not have any such mechanism. So if TCP detects congestion and reduces its packet transmission it might get starved for

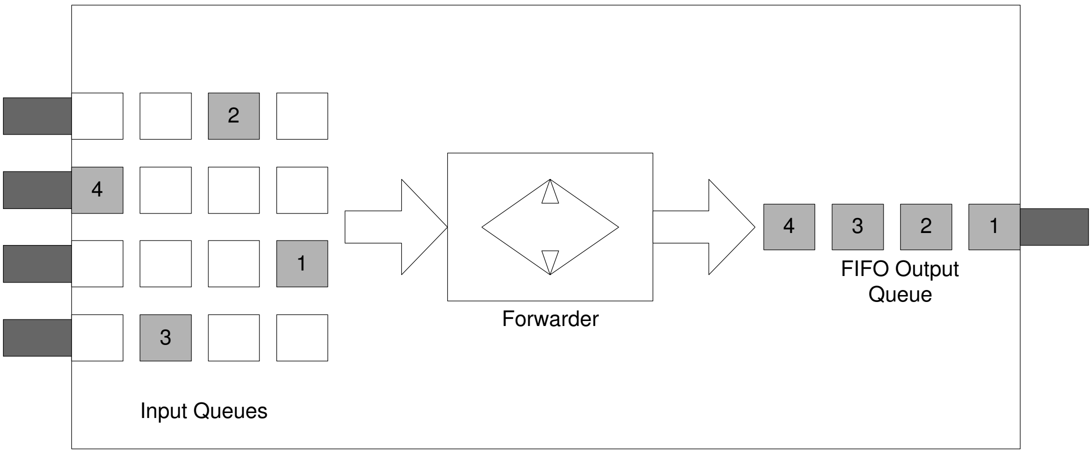


Figure 1-1: FIFO queuing

resources as UDP will not adjust and might start using more bandwidth. In a scenario where the transmission capacity exceeds the network's admission capacity this form of queuing is ideal. The queuing is required just to make sure that the transient effect, like a sudden increase or burst of traffic does not cause packet discard. The queuing is also required to resolve contention in case of simultaneous packet arrival.

### Priority queuing

This was the first queuing to become popular after FIFO. Priority queuing is based on the classification of the packet into a service class. Based on the classification, it is placed into the appropriate queue. The scheduling of the queue is very simple; a lower priority queue would not be serviced until there are no packets in higher priority queues [1].

Administrators need to be careful while using this queuing. A small error in judgment can cause very poor deployment of this scheme, as any traffic in the higher priority queue will delay the traffic in the lower priority queue. So it is a common practice to keep the higher priority queues shorter and the low priority queue longer.

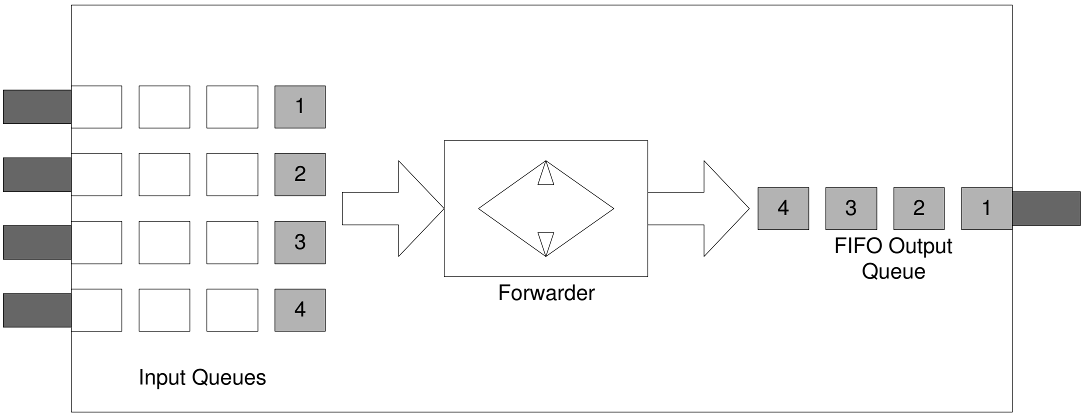


Figure 1-2: FIFO queuing with contention

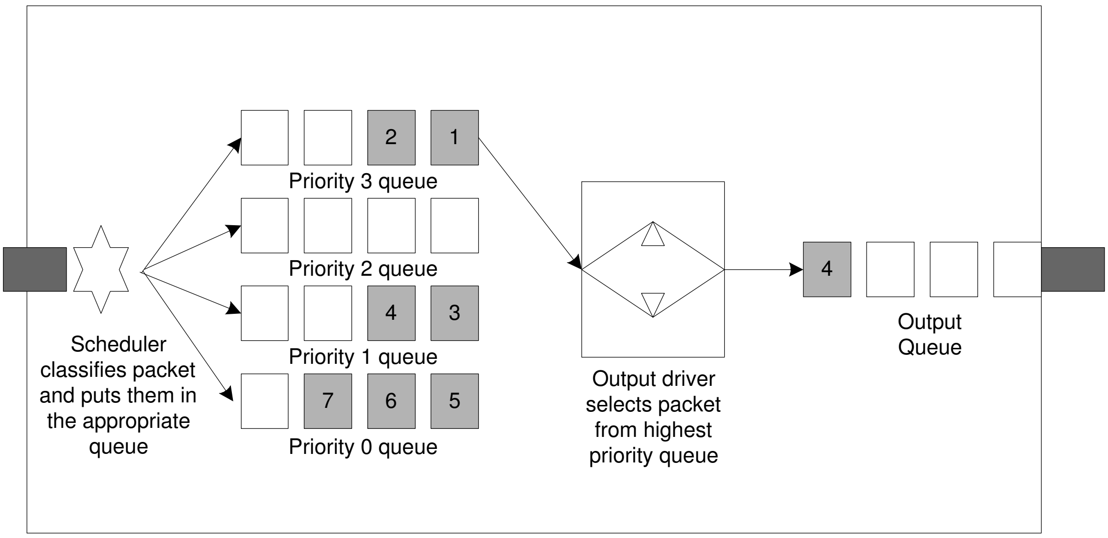


Figure 1-3: Priority queuing

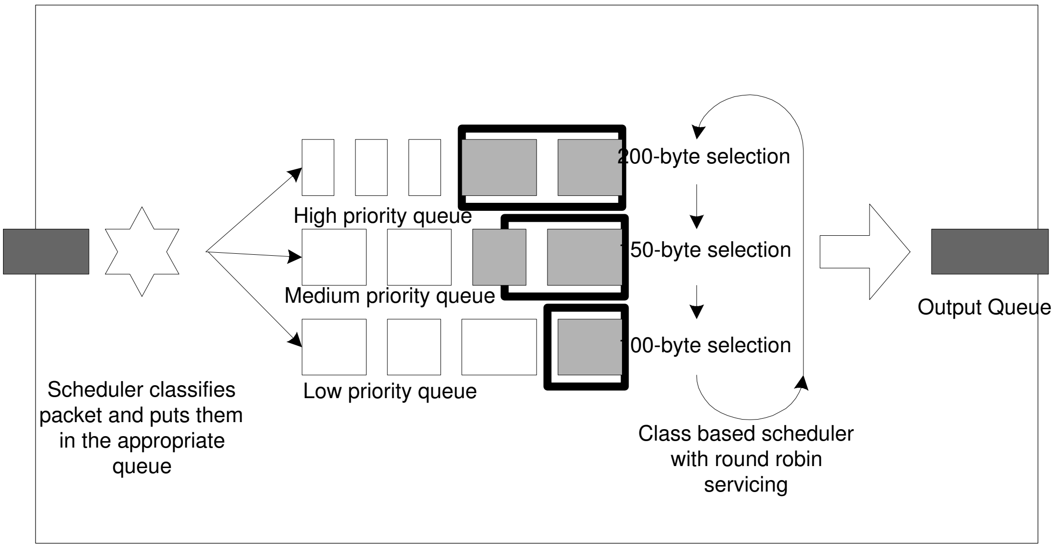


Figure 1-4: Class based queuing

Another pitfall here is incorrectly assuming that a larger number of queues provides better granularity. In fact, the more queues there are, the more difficult it becomes to differentiate the traffic into higher classes without affecting the low priority queues. So the general tendency is to keep the number of queues to the minimum possible, usually no more than three.

**Class based queuing (CBQ)**

Unlike priority queueing, class based queuing, also known as custom queuing or weighted round robin (WRR) tries to avoid the complete denial of service of the low priority traffic. In this type of queuing, administrators can specify the length of the queue and the amount of data to be drained from the queue in every iteration. This guarantees at least one packet from each queue in every iteration so that there is no denial of service to any queue. The danger in this type of queuing it that the lower priority queue might actually get a higher level of service. For example, consider

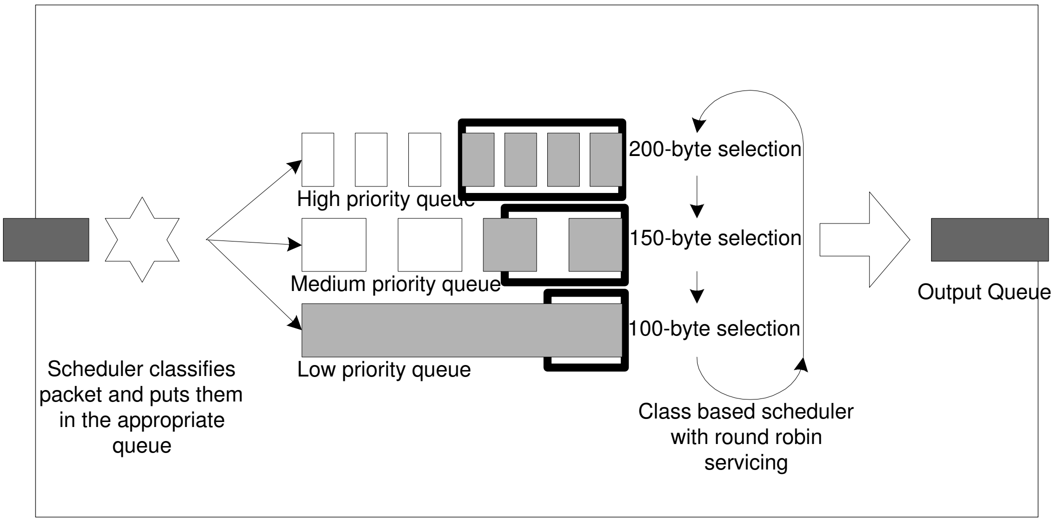


Figure 1-5: CBQ with lower priority getting higher level of service

three queues with the highest queue passing 200 bytes, the middle passing 150 bytes and the lowest queue passing 100 bytes of data in each pass. If the typical packet size in the lowest priority queue is 1550 bytes and in the highest priority queue is 60 bytes, then even if the device sends one packet from the lowest queue it still takes up more service than it should. A solution to this is to increase the size of the data to be passed in each iteration for all the queues. This will include the maximum size of the packet and a more accurate distribution of resources. But there is a flip side to this solution; it includes a lot of delay in a single packet stream, especially if the device is not of high speed (such as optical). For high-speed devices this scheme works well, as each round finishes much faster.

### Weighted Fair Queuing (WFQ)

Weighted Fair Queuing [1] tries to schedule packets in a fair manner based on the weights given to each stream of traffic. In this case two flows having small packet sizes can be differentiated based on the weights they have. So that if packets such as telnet and voice are using small packet sizes, voice can be given higher weight so that it gets priority over the telnet packets. Individually the flows are considered as FIFO. This is very similar to class based queuing. If the bandwidth is not being utilized entirely, then it is distributed among other flows based on the weights.

All queuing mechanisms have their advantages and disadvantages. FIFO queuing is good for large queues and fast-switching environments with predictable outcomes. But they do not implement any service policy. In priority queuing the highest priority traffic receives low jitter and low packet loss. This type of queuing implements service policies. The main difference between class based queuing and priority queuing is that CBQ offers at least some level of service to the low priority queues. Weighted fair queuing offers dynamic allocation of resources to all queues based on the weights configured. This prevents any single uncontrolled application from bursting traffic into the network. Queuing by itself cannot provide the entire QoS provisioned solution. All the other network and device aspects need to be configured appropriately for a solution to work.

### 1.3.4 Congestion Management

Congestion management can be defined as the ability of a network to effectively deal with heavy traffic volumes [5]. This aspect of QoS is more than just a single entity and mostly encompasses more than one tool to make it work. For example, differential congestion management is when a lower-precedence class has a higher probability of being dropped and additional buffer space is only available to traffic of higher class.

Queuing can be one of the tools which implement congestion management for the router and the network. For example, additional congestion management techniques can be combined with weighted queuing to give higher preference to higher class of traffic to create a congestion management solution.

## 1.4 DiffServ: Algorithm developed with QoS in mind

Differentiated Services(DiffServ) achieves scalability by aggregating traffic classification into categories. This information is passed through IP-layer packet marking using the DiffServ (DS) field, in place of the Type Of Service (TOS) field. Based on this information, packets will receive a particular per-hop forwarding behavior by the routers along their path. The Differentiated Services model may be implemented from the host or at an edge router. Also, extensions to Multi Protocol Label Switching (MPLS) have been created to associate per-hop-behaviors with Label Switched Paths(LSPs). This is a domain-based technology so in a network we can have a set of devices that form a smaller network that uses DiffServ. That is why there are ingress, egress and intermediate routers for DiffServ.

DiffServ is very widely used in attaining the QoS goals of classification and eventually differential behavior.

## 1.5 Summary

In this chapter we discussed the basics of QoS. With this information we can now look into the performance metrics that need to be measured. In the next chapter we identify the metrics and discuss the ways to measure those along with the parameters to be changed such as classification and remarking.

# Chapter 2

# Methodology Development

The goal of this thesis is to develop methods of testing aspects of QoS and investigating the behavior of devices when subjected to testing. To develop the methods we first need to identify the performance metrics that are most important from the QoS point of view.

## 2.1 Identifying performance metrics

Differentiation or classification, admission control and queuing are the most important aspects to consider while configuring the network for QoS. Measurement might be in terms of checking the validity or expected behavior. For example, when it comes to admission control, administrators need to just check whether a device is capable of policing or dropping the packets based on set criteria. However, such easily observed behavior is rare. More frequent and often more interesting, are the subtle variations in behavior of a device when observed while sending traffic through it. But we also need to look at the QoS metrics which get affected by classification and queuing. If traffic is sent without any classification and then the same type of traffic is sent with some, the first change we can look for is how much extra time it takes for the packet to travel in the latter case. The measurement here is known as latency and is defined as the time taken by a packet to travel from the source to the destination. Because of classification there might be a change in the throughput as well.Throughput is defined as the amount of data tranferred from the source to the destination or processed in a specific amount of time. Now when we consider queuing, looking at similar measurements might indicate the effectiveness of the queuing on a device. Similarly, jitter, which is the difference in latency of packets, appears relevant when it comes to queuing.

So the metrics identified to be measured are:

**Throughput:** typically measured in percentage of available bandwidth, bits or bytes per second, frames per second.

**Latency:** typically measured in milliseconds or microseconds.

**Jitter:** typically measured in microseconds or nanoseconds.

We will also need to concentrate on the effect of different configurations of the device on these metrics. So the first step will be to make sure that we have resources to measure these metrics.

A standard Generator/Analyzer is capable of measuring many of these values. Devices that are available at UNH InterOperability Laboratory (UNH-IOL) are:

- Ixia 1600T
- Navtel Interwatch
- Spirent Adtech AX/4000
- Spirent Smartbits 600

All of the above devices are capable of behaving as a traffic generator as well as analyzer. They are quite similar in their capabilities from the thesis perspective. They are capable of taking user input through a GUI-based client as well as scripts. A new form of scripting becoming popular nowadays is what we term GUI-based

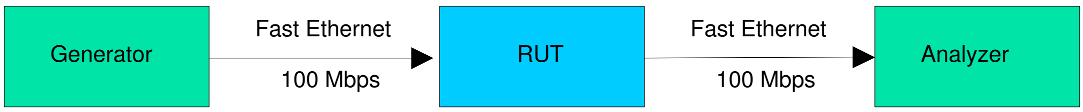


Figure 2-1: Experimental setup diagram

scripting. In this you can use the GUI to put a few tests in a pipeline (similar to a script) and leave them to run unattended. Sometimes it is a non-trivial issue to measure values such as latency and jitter. One needs to be clear about the capabilities of the analyzer especially what it can measure and what needs extra work to get the desired measurements.

## 2.2 Experimental Setup

Figure 2.1 shows the setup for the experiments described in Chapter 3 where the generator and analyzer are connected to the Router Under Test (RUT) via a fast-ethernet link running at 100 Mbps. Spirent Smartbits 600 was used as the generator and analyzer.

Queuing is one of the aspects that, if chosen correctly, could contribute greatly to the success of the QoS solution. Different priority traffic placed in different queues will have the effect of prioritization accordingly. But to place the traffic into a specific queue we first need to classify the traffic. So classification is one of the basic functions performed on a packet when it arrives at a QoS enabled device. This aspect is then followed by remarking if required. As classification and remarking are the basic building blocks for QoS, we look into test cases with combination of these two aspects.

### Case 1: No classification and no remarking

This case acts as a reference or baseline case for other experiments. The router allows traffic to pass through with no classification and no remarking.

### Case 2: Classification and no remarking

IPv4 TCP or UDP traffic is sent from the generator to the analyzer in this case, when there is classification of packets done at the RUT, i.e., the router looks into the packets to categorize them into classes. This creates extra work for the router on a per packet basis.

To deliver superior service to a class of traffic, the first step is to differentiate between the packets. This process segregates the packets that are supposed to receive better treatment from the router. Here we send only one stream of traffic and it is classified into a high priority stream. The goal is to make the device work to classify the traffic but not to classify one stream from another. We are concentrating on the extra work put on the device by enabling classification and its effect on the performance.

### Case 3: Classification and remarking

In this test case traffic is sent from the generator to the analyzer, while classification and remarking of packets is performed at the RUT. By classification and remarking, we mean that the router looks into the packets to categorize it into a class and based on the classification the packet will have the TOS or DiffServ Code Point (DSCP) [7] field in the packet changed. This creates even more work for the router to do on a per packet basis, as opposed to the previous test case.

Remarking is commonly performed at the edge of a network. When a packet crosses from a network under one administration to a network under another, typically it will get remarked. This is because every network has its own understanding of values in the header field, such as type of service. When a packet arrives into a network it will be remarked in accordance to the set of rules implemented by the management of that network. Similarly there might be some agreements with the neighboring networks, so that when a packet leaves from the network into one of the neighbors, the last router might want to remark it based on the agreements. Also, if there are no agreements the last node might still remark the packets, so that no information about this network and its packet management is revealed to the neighboring network. Typically in this case it will be remarked to one default value.

## 2.3 Throughput Measurement

It is amount of data transferred from the source to the destination or processed in a specified amount of time. Throughput measurement gauges the amount of data transferred from one place to another or processed in a specified amount of time. Data transfer rates for disk drives and networks are measured in terms of throughput. Throughputs are typically measured in bits per second (bps), bytes per second (Bps) and frames per second (fps).

In the initial set of tests, throughput is measured in terms of data sent and data received. The readings will therefore not account for Ethernet frame headers and will just measure the Ethernet payload. This is the reason why the maximum percentage of data that can be sent across in 64 bytes frame is much less than 1518 bytes. This is also why the bandwidth utilization will not reach 100% for the throughput graphs.

## 2.4 Latency Measurement

Latency is defined as the time taken by a packet to travel from the source to the destination [5]. It could be considered as propagation delay for a packet from the source to the destination without the packetization delay at the generator or the analyzer. From a real network perspective this measurement is very difficult to obtain, as it posses the problems of clock synchronization, since in most cases the source and destination are physically far apart. To avoid this, latency has been often measured in terms of round trip time and then divided by two. But when we are talking about our experimental setup, latency can be measured in terms of time from the source to the destination, since in our case the source and the destination are not physically far apart and can use the same system clock. These tests measure the latency, i.e., the time taken by a packet to travel from the generator to the analyzer. Here the traffic generator is the same as the traffic analyzer for measurement of the values so that the clock used to measure the time is the same and we do not have to deal with clock synchronization issues.

## 2.5 Jitter Measurement

Jitter is the variation in the latency of packets at the destination. If the jitter value is high the performance in some time-sensitive applications, such as voice over IP, might get affected. Excessive jitter might cause pops or clicks in audio communications [5].

There are several ways of measuring jitter based on parameters being taken into account while measuring. Following are the four ways of measuring jitter.

### 2.5.1 Method 1: Difference in maximum and minimum latency

Maximum jitter is the difference in the maximum and minimum latency measured over a testing interval. This measurement is easy to perform and most test equipment available today provide maximum and minimum latency measurements while testing latency. The most common calculation is as follows.

$$Jitter = Maximum \quad Latency - Minimum \quad Latency$$

Here, the latency measurements are over the entire test duration.

### 2.5.2 Method 2: Difference in time of arrival of consecutive packets

In this method, jitter is calculated as the change in the difference in the arrival time of the packet [9]. This method of measuring jitter takes into account, the arrival time and not the latency.

$$\Delta arrival_n = |Arrival_n - Arrival_{n-1}|$$

where, $n$ is the current packet.

$$Jitter_n = |\Delta arrival_n - \Delta arrival_{n-1}|$$

where, $n$ is the current packet.

### 2.5.3 Method 3: Difference in latency of consecutive packets

In this method, jitter is measured as difference in the latency of current packet and the previous packet [10].

$$Jitter_n = |Latency_n - Latency_{n-1}|$$

where, $n$ is the current packet.

If we express latency as,

$$Latency_n = Arrival_n - Departure_n$$

Then jitter can be written as,

$$Jitter_n = |(Arrival_n - Departure_n) - (Arrival_{n-1} - Departure_{n-1})|$$

can be re-arranged as,

$$Jitter_n = |(Arrival_n - Arrival_{n-1}) - (Departure_n - Departure_{n-1})|$$

Looking at the above formula we can see that jitter measured by this method relates closely to method 2. The main change being the difference in the departure time is taken into account, so that, even if the generator is not generating packets at constant rate, it doesn't introduce any error in the measurements.

### 2.5.4 Method 4: Difference in current and average latency

In this method, jitter is calculated as difference in the latency of current packet to the average latency. Here average latency is calculated over the duration of the test. This method was suggested and endorsed by a service provider.

$$Jitter_n = |Latency_n - AverageLatency|$$

where, $n$ is the current packet.

When jitter is measured taking into account only the arrival time, it is called one-point measurement. In this case it is not required to measure latency of a packet. Method 2 would be an example of this type.

But when we look into departure as well as arrival time of a packet to calculate the jitter it is known as two-point measurement. Method 2, 3 and 4 are examples of two-point measurements.

## 2.6 Jitter measurement examples

Now we will look into some examples and see how different methods measure jitter under different scenarios. In the following examples, six packets are assumed to leave the source at times: 0 ms, 2 ms, 4 ms, 6 ms, 8 ms and 10 ms.

The experimental setup is same as shown in Figure 2.1. The arrival time of each packet will be different for different scenarios and are listed in the examples accordingly.

**Case 1: Constant Latency**

The arrival time of the six packets at the destination are: 2 ms, 4 ms, 6 ms, 8 ms, 10 ms and 12 ms.

This case shows a network behaving in very consistent manner in terms of latency. All the packets have same latency. This is an ideal case and acts as a baseline for reference. Shown in Figure 2.3(a).

**Case 2: Alternating Latencies - Saw tooth**

The arrival time of the six packets at the destination are: 1 ms, 5 ms, 5 ms, 9 ms, 9 ms and 13 ms.

This case shows the network introducing two different latency values for consecutive packets. This kind of behavior could be seen in a traffic-engineered network where alternate packets take alternate paths or a router introduces alternating latency because of packets being placed in different queues. Shown in Figure 2.3(b).

**Case 3: One packet having high latency - Spike**

The arrival time of the six packets at the destination are: 1 ms, 3 ms, 11 ms, 7 ms, 9 ms and 11 ms.

This case shows one packet getting very high latency as opposed to the rest. This kind of behavior is a common occurrence in networks. Shown in Figure 2.3(c).

**Case 4: Latency changes once to a different value - Step**

The arrival time of the six packets at the destination are: 1 ms, 3 ms, 5 ms, 9 ms, 11 ms and 13 ms.

This case shows the network introducing one latency value to some traffic and then another for the rest. This kind of behavior can be seen if there is some change

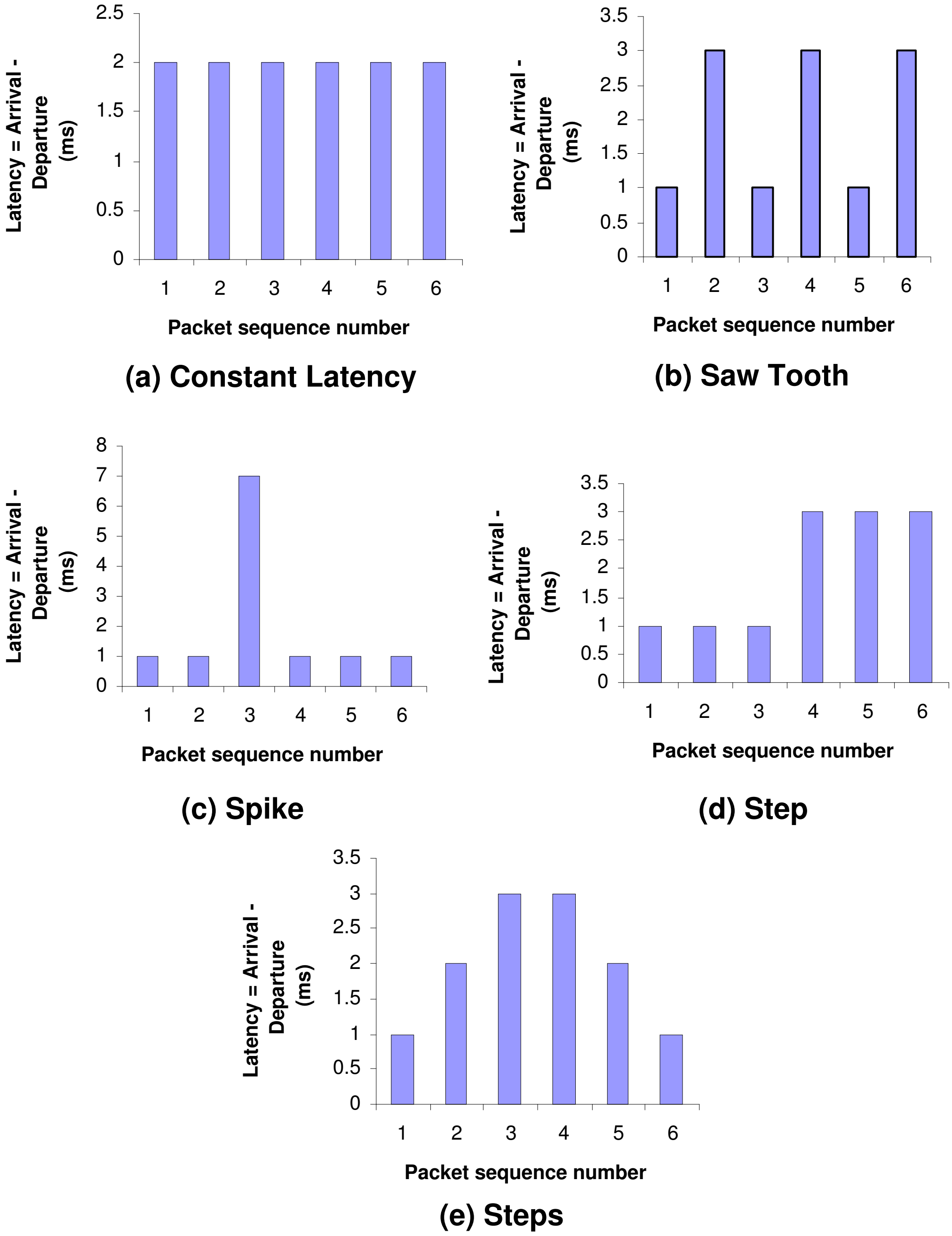


Figure 2-2: Cases showing latency

in the network or change from off peak to peak hours. Shown in Figure 2.3(d).

**Case 5: Latency gradually increasing and decreasing - Steps**

The arrival time of the six packets at the destination are: 1 ms, 4 ms, 7 ms, 9 ms, 10 ms and 11 ms.

This case shows the network latency value increasing gradually and then decreasing. Here the change in latency is by same magnitude for consecutive packets. This kind of behavior could be seen in a network which is not stable. Shown in Figure 2.3(e).

Table 2.1 shows jitter calculations for all the methods for different cases. It also shows average latency and jitter in the bottom along with maximum jitter. Method 1 has been deliberately placed last to compare it with other methods from a jitter distribution perspective. The table also includes standard deviation for the latency values.

### 2.6.1 Method Analysis

Method 1 takes the difference between the maximum and minimum latency. This is an easy way of measuring jitter and produces fairly acceptable values, but if a network has spikes as shown in case 3 then the jitter value does not reflect the actual behavior. For all other cases method 1 gives dependable measurements.

Method 2 only takes the current and previous two arrival times for the measurement and not the latency. Hence the effect of spike does not propagate through the entire distribution. However, this method gives too large values for saw tooth type of measurements as shown in case 2. Also it gives quite low values for steps as shown in case 5.

Method 3 compares the current and previous packet latency . This method gives low jitter values for step change in latency as shown in case 4. This method does

| Case 1: | Departure | Arrival | Latency | Method 2 | Method 3 | Method 4 | Method 1 | Std. Dev. |
|---|---|---|---|---|---|---|---|---|
| | 0 | 2 | 2 | | | 0 | 0 | 0 |
| | 2 | 4 | 2 | | 0 | 0 | | |
| Constant | 4 | 6 | 2 | 0 | 0 | 0 | | |
| | 6 | 8 | 2 | 0 | 0 | 0 | | |
| | 8 | 10 | 2 | 0 | 0 | 0 | | |
| | 10 | 12 | 2 | 0 | 0 | 0 | | |
| | | **Average** | **2** | **0** | **0** | **0** | | |
| | | | **Maximum** | **0** | **0** | **0** | | |

| Case 2: | Departure | Arrival | Latency | Method 2 | Method 3 | Method 4 | Method 1 | Std. Dev. |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | | | 1 | 2 | 1.095445 |
| | 2 | 5 | 3 | | 2 | 1 | | |
| | 4 | 5 | 1 | 4 | 2 | 1 | | |
| SawTooth | 6 | 9 | 3 | 4 | 2 | 1 | | |
| | 8 | 9 | 1 | 4 | 2 | 1 | | |
| | 10 | 13 | 3 | 4 | 2 | 1 | | |
| | | **Average** | **2** | **4** | **2** | **1** | | |
| | | | **Maximum** | **4** | **2** | **1** | | |

| Case 3: | Departure | Arrival | Latency | Method 2 | Method 3 | Method 4 | Method 1 | Std. Dev. |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | | | 1 | 6 | 2.44949 |
| | 2 | 3 | 1 | | 0 | 1 | | |
| | 4 | 11 | 7 | 6 | 6 | 5 | | |
| Spike | 6 | 7 | 1 | 4 | 6 | 1 | | |
| | 8 | 9 | 1 | 2 | 0 | 1 | | |
| | 10 | 11 | 1 | 0 | 0 | 1 | | |
| | | **Average** | **2** | **3** | **2.4** | **1.666667** | | |
| | | | **Maximum** | **6** | **6** | **5** | | |

| Case 4: | Departure | Arrival | Latency | Method 2 | Method 3 | Method 4 | Method 1 | Std. Dev. |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | | | 1 | 2 | 1.095445 |
| | 2 | 3 | 1 | | 0 | 1 | | |
| | 4 | 5 | 1 | 0 | 0 | 1 | | |
| Step | 6 | 9 | 3 | 2 | 2 | 1 | | |
| | 8 | 11 | 3 | 2 | 0 | 1 | | |
| | 10 | 13 | 3 | 0 | 0 | 1 | | |
| | | **Average** | **2** | **1** | **0.4** | **1** | | |
| | | | **Maximum** | **2** | **2** | **1** | | |

| Case 5: | Departure | Arrival | Latency | Method 2 | Method 3 | Method 4 | Method 1 | Std. Dev. |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 1 | | | 1 | 2 | 0.894427 |
| | 2 | 4 | 2 | | 1 | 0 | | |
| | 4 | 7 | 3 | 0 | 1 | 1 | | |
| Steps | 6 | 9 | 3 | 1 | 0 | 1 | | |
| | 8 | 10 | 2 | 1 | 1 | 0 | | |
| | 10 | 11 | 1 | 0 | 1 | 1 | | |
| | | **Average** | **2** | **0.5** | **0.8** | **0.666667** | | |
| | | | **Maximum** | **1** | **1** | **1** | | |

Table 2.1: Comparison of different methods for each case

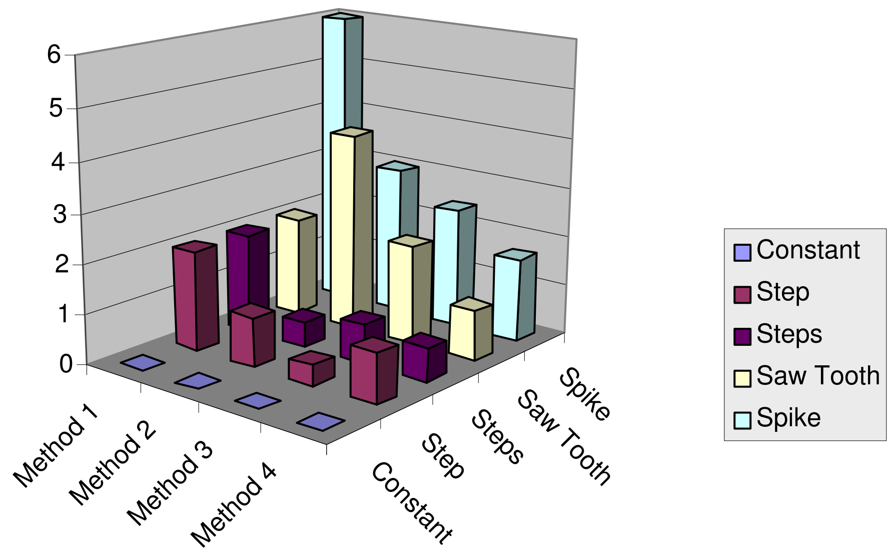


Figure 2-3: Method and cases comparison

propagate the spike effect for the entire distribution.

Method 4 compares current latency with the average latency. In this method, irregularities such as spikes get averaged in and the affect the entire distribution. This method gives low jitter values for saw tooth latency while giving continuous jitter for step change in latency. These results are summarized in a Figure 2.4.

In the tests performed for this thesis we used method 1 as it was supported by the generators/analyzers that we have. For method 3 and 4 we need to measure per packet latency which is not supported. Method 2 does not take into account the latency of a packet. Looking at these aspects, method 1 worked out as the solution for jitter measurement. Method 1 is also the most widely accepted method for jitter.

## 2.7 Generator and Analyzer

To have faith in the values measured we first need to have a level of trust in the generator/analyzer. To achieve that trust, some experiments were performed to verify that the traffic generator is able to generate traffic exactly as it claims to generate.

We used Symbol Generator(Symbgen) to measure against the generator/analyzer. The Symbgen is a Fast Ethernet encoder/decoder, that allows us to transmit and receive at the 5-bit layer. Physical Coding Sub-layer (PCS) converts a 4-bit nibble from the higher layer into a 5-bit code group to send over the physical medium. This allows for the addition of things such as idle code groups to maintain synchronization, combination of two control code groups to make up a Start of Stream Delimiter (SSD), and combination of two other code groups to indicate the End of Stream Delimiter (ESD). Symbgen allows us to send any combination of valid as well as invalid symbols. It supports the reception of a stream containing an SSD, including the time-stamping of the reception. With this we have control over such things as how many nibbles of idle we wish to transmit between a number of frames. It has been used for the testing of Fast Ethernet equipment here in the lab for over eight years. It was originally developed by Digital Technology Inc. as a Fiber Distributed Data Interface (FDDI) / Copper Distributed Data Interface (CDDI) testing tool and modified by IOL for use as a Fast Ethernet testing tool. Fast Ethernet and FDDI/CDDI use a similar physical layer and encoding scheme.

Following are the tests that were performed on the generator/analyzer. The topology setup for the tests is shown in Figure 2.5

### 2.7.1 Line rate traffic generation

Here we test whether the generator is able to generate line rate traffic which in this case is 100Mbps. Transmitting frames at 100Mbps means that the Inter-Frame

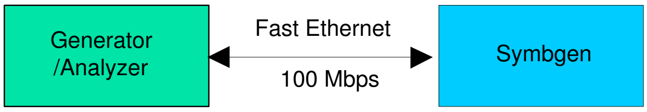


Figure 2-4: Generator/Analyzer tested against Symbgen

Gap(IFG) should be a constant value of 0.96 $\mu$s. The generator is configured to send line rate traffic and Symbgen is used to analyze the frames and check whether the IFG is a constant, 0.96 $\mu$s.

### 2.7.2 Line rate marked traffic generation

This test is similar to the previous one but with the difference that the packets generated are marked with a non-zero ToS field in the packet. This test is done to verify that the generator can send out packets with ToS field marked at 100Mbps. Symbgen is used to analyze the frames and check whether the IFG is a constant, 0.96 $\mu$s.

### 2.7.3 Greater than line rate traffic reception

This test is to check the resiliency of the generator/analyzer. Symbgen can generate frames with IFG of less than 0.96 $\mu$s. With this feature we can check whether the analyzer is able to handle the reception of this traffic stream or not. Here the Symbgen was configured to send out traffic stream with IFG of 0.88 $\mu$s.

### 2.7.4 Specific Inter-Frame Gap

This test verifies whether the generator sends out frames with specified IFG or not. In this case the streams are configured to have an IFG of 100 $\mu$s in the first case, 150 $\mu$s in the second case and 200 $\mu$s in the third. With the help of Symbgen we calculate

| Frame Size(bytes) | Percentage |
|---|---|
| 58 | 55% |
| 62 | 2% |
| 594 | 20% |
| 1518 | 12% |
| Random | 11% |

Table 2.2: Imix distribution

the IFG of the received frames. It should match with the values set at the generator.

### 2.7.5 Results of Generator/Analyzer testing

All the generators/analyzers passed every test except one, where the device failed tests line rate constant inter-frame gap tests (2.7.1 and 2.7.2). In these tests the inter-frame gap for 64 bytes at 100Mbps, which is suppose to be 0.96 $\mu$s, was observed to be alternating between 0.96 and 1.04 $\mu$s. After consultation with the test equipment manufacturer, the fluctuation was attributed to the hardware used for the device. This device was not utilized for testing in this thesis.

## 2.8 Internet mix traffic tests

Internet mix (Imix) is a traffic distribution of frame sizes which closely relates to real internet traffic. There are a few Imix distributions available with most agreeing on a broad perspective. Based on a few sources [12][13], the distribution used for the tests is shown in Table 2.2.

Here the frame size includes the 14-byte layer 2 header and 4-byte CRC. For Ethernet the frame sizes in the distribution that are below 64 byte are padding up to

64 byte. The test were repeated to vary the percentage of bandwidth utilized from 10% to 95%.

## 2.9 Effect of packet size on latency

It is important to find out how the packet size affects the latency measurement. We will need to discuss how latency gets affected when a 64-byte frame is measured for latency as opposed to a 1518-byte frame. The generators/analyzers used to measure the values for latency measure the time taken from the point when a packet totally leaves the generator until the packet totally arrives at the analyzer. This way the frame size does not effect the latency measurement. In other words, the latency measurement does not include the packet's propagation delay.

## 2.10 Confidence Level of Data

There has to be some level of confidence in the data recorded from the experiments. For example, to have a confidence level of 95%, the formulae below help us ascertain the validity.

$$Standard \quad Deviation(\sigma) = \frac{\sqrt{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}}{n(n-1)}$$

where $n$ is the number of samples and $x$ is the value of a sample. Then,

$$Confidence \quad Interval = 1.96 * \frac{\sigma}{\sqrt{n}}$$

The constant 1.96 in the above equation has been looked up in the normal distribution table for a confidence level of 95%. The confidence level indicates that with 95% assurance we can say that the output value will be within the average confidence interval.

All the experiments were repeated for $10(n)$ iterations. The confidence level of the data came out to be at least 95%. The number of iteration were fixed at 10 in order to keep the number of experiments at a predictable level.

Confidence level does not apply to Imix tests. This is because Imix tests have a random frame size in the distribution and that random percentage makes the number of frames different for different test runs.

# Chapter 3

# Experiments

The experiments performed below are based on the methodology developed in Chapter 2.

The results have been classified on the basis of vendors A, B, C and D. All the devices used belong to the category of devices which are currently deployed and being used as service provider edge equipment.

## 3.1 Vendor A: Throughput

### 3.1.1 Case 1: No classification and no remarking

Figure 3.1 shows the graph of unclassified and not remarked traffic as it is passed through the router with carried load versus offered load. As expected for all the frame sizes the graph is straight line, showing the sent data being same as the received data.

For the throughput experiments Spirent Smartbits 600 was used to send IP traffic, i.e., no TCP or UDP.

For throughput experiments it is also interesting to see offered load on the x-axis and carried load on the y-axis in frames per second, as it shows the performance of a device with respect to frames it can handle per second. Figure 3.2 also shows multiple straight lines. This graph can be used to compare other graphs plotted in frames per second in this chapter.

The Imix tests done here are plotted for throughput in frames per second. The

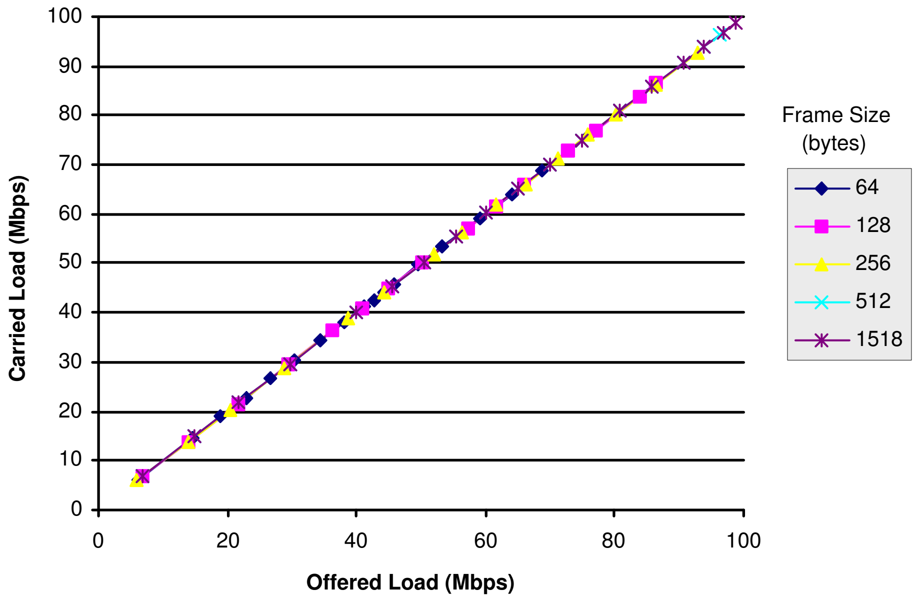


Figure 3-1: Vendor A: Throughput with no classification and no remarking

latency and jitter will not be comparable because of the mixture of different packet sizes especially with random length packets which will not match for all the cases. Figure 3.3 shows the graph of Imix traffic with no classification and no remarking. The graph is a straight line showing the offered load same as carried load.

### 3.1.2 Case 2: Classification and no remarking

Figure 3.4 shows the graph of throughput with classification and no remarking. It has data sent on the x-axis and data received on the y-axis. As in the baseline graph typically we expect the graph, to be a straight line, showing offered and carried load to be the same. But looking at the 64-byte frame size we can see that the data received drops after the initial straight line.

Figure 3.5 shows throughput graph with carried load in frames per second on the

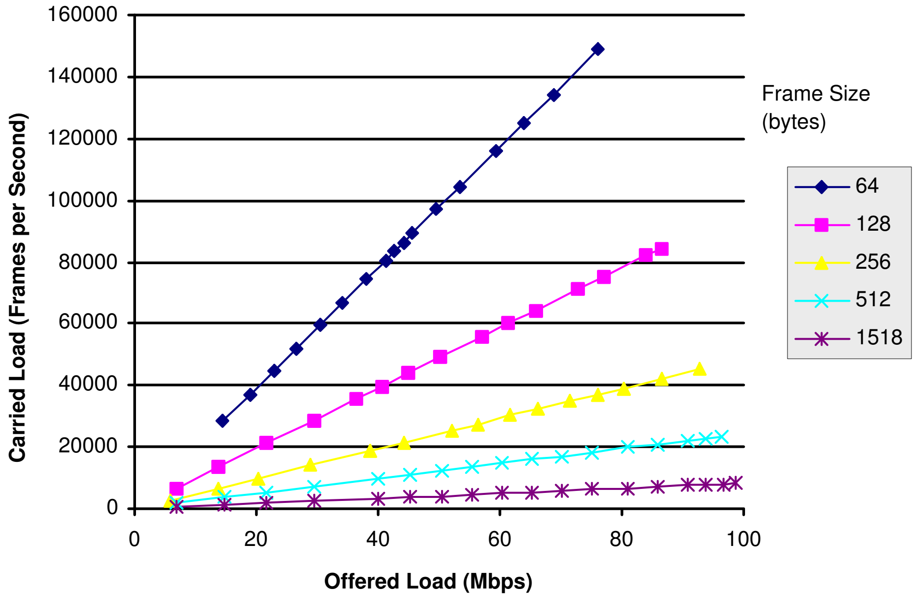


Figure 3-2: Vendor A: Throughput in frames per second with no classification and no remarking

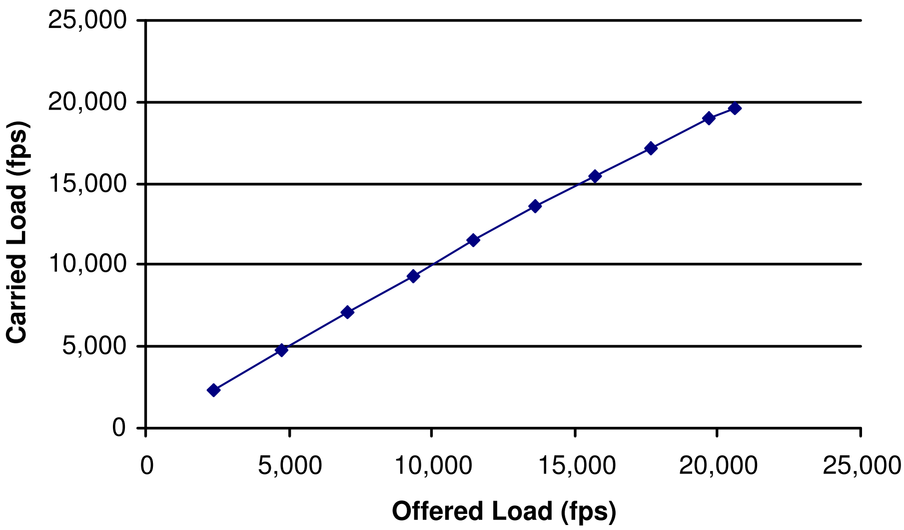


Figure 3-3: Vendor A: Imix traffic with no classification and no remarking

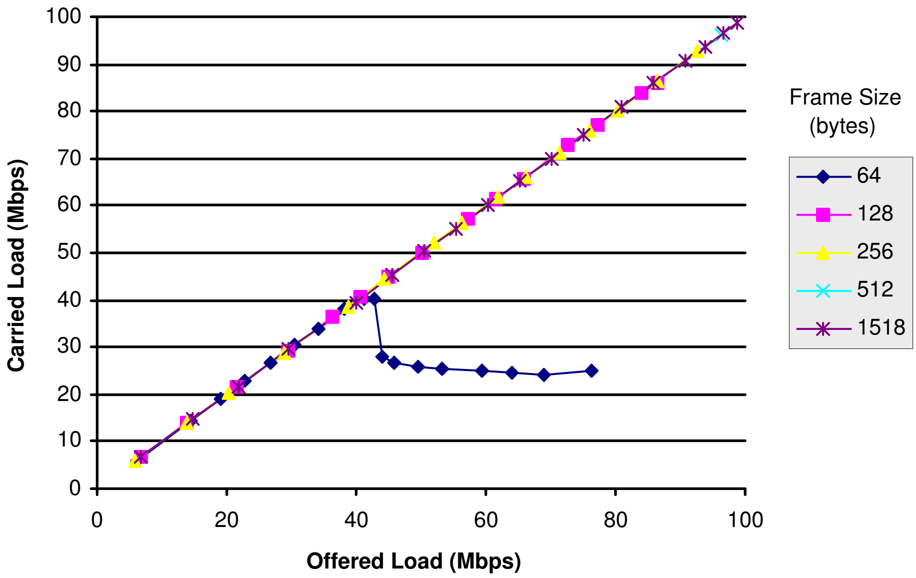


Figure 3-4: Vendor A: Throughput with classification and no remarking

y-axis. It can be observed that after a certain offered load the 64-byte frames carried, drops, showing that the device could no longer maintain the carried load at the offered value.

Figure 3.6 shows the throughput graph for Imix traffic with classification and no remarking. We can also see the performance deterioration for this case. At around 14,000 fps the carried load saturates.

### 3.1.3 Case 3: Classification and remarking

Figure 3.7 shows the graph of throughput with classification and remarking enabled on the device. It has data sent on the x-axis and data received on the y-axis. In an ideal case the graph is expected to show 1:1 correspondence between offered load and carried load. However it can be observed that for smaller frame sizes the performance goes down and this is more pronounced than what it was for the previous case.

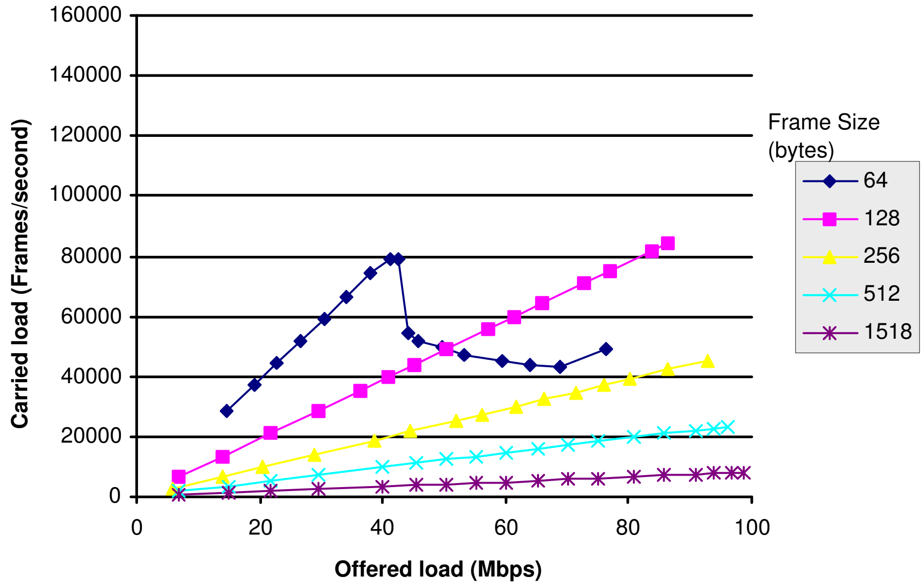


Figure 3-5: Vendor A: Throughput in frames per second with classification and no remarking

Figure 3.8 shows the same results but with carried load measured in terms of frames per second. As can be seen, 40000 frames per second seem to be the throughput of the packet processor of the RUT with classification and remarking. The performance for 64, 128 and 256 byte frame size starts dropping at around 40000. Figure 3.9 shows throughput graph for the Imix traffic with classification and remarking. We can also see the performance deterioration for this case. At around 13,500 fps the carried load saturates.

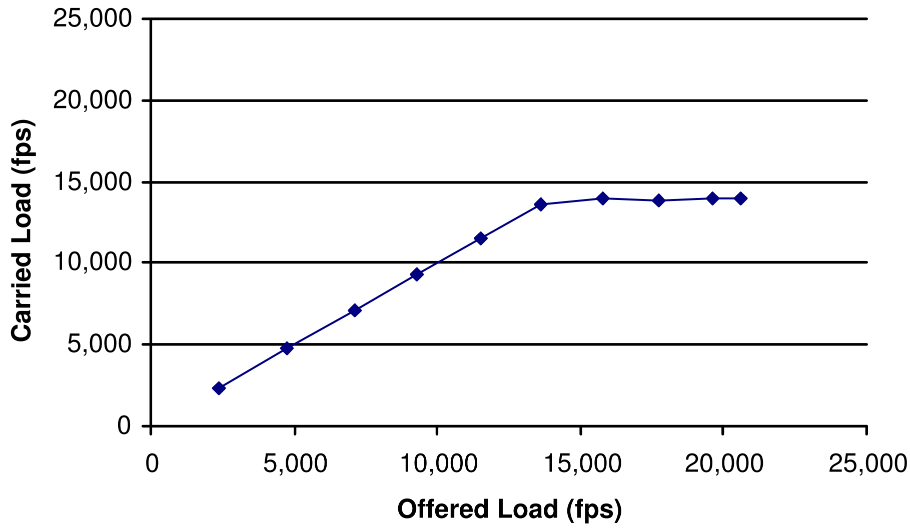


Figure 3-6: Vendor A: Imix traffic with classification and no remarking

## 3.2 Vendor A: Latency

### 3.2.1 Case 1: No classification and no remarking

Figure 3.10 shows average latency values against percentage bandwidth used. In this case the router was not classifying or remarking any packets. This case acts as a baseline and helps us compare performance with other scenarios. For the experiments performed in this section and later, TCP packets were sent across the network through the Spirent Smartbits 600 which utilizes a 12-byte time stamp as data, because of which the smallest frame size is 76 bytes. The difference in the latency among different frame sizes might be because of the delay in propagation. The analyzer receiving the packet could be waiting for the entire packet to arrive, hence increasing the latency of higher frame sizes. This issue needs to be investigated with the analyzer vendors.

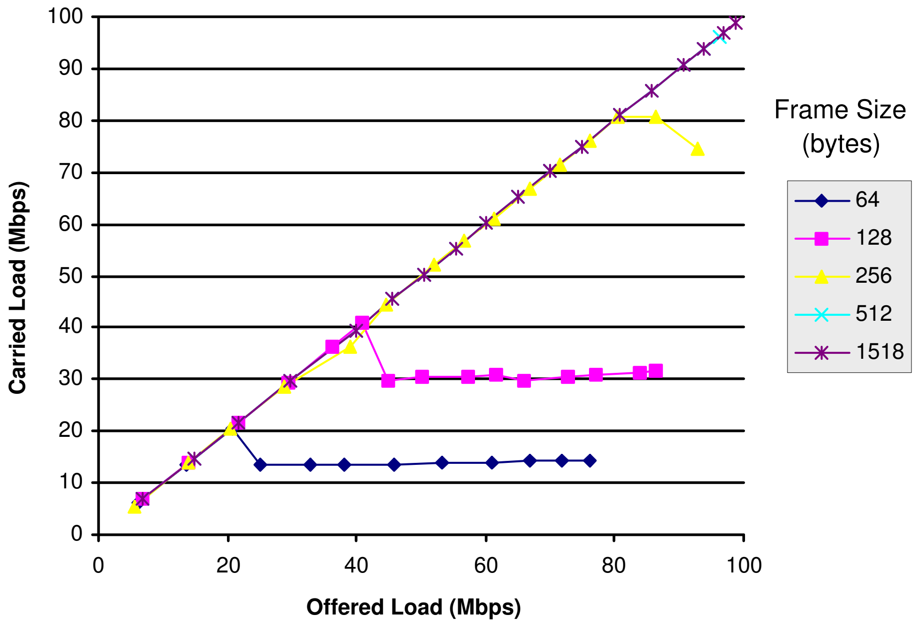


Figure 3-7: Vendor A: Throughput with classification and remarking

### 3.2.2 Case 2: Classification and no remarking

Figure 3.11 shows line utilization versus latency for classification and no remarking. Only the 76-byte frame size show significant increase in latency as the line utilization increases when compared to Figure 3.7.

Figure 3.12 shows the maximum, minimum and average latency for 76-byte frame size.

### 3.2.3 Case 3: Classification and remarking

Figure 3.13 shows latency versus line utilization with classification and remarking. There is a significant increase in the latency of the 76-byte frame in this case as well. Figure 3.14 shows the maximum, minimum and average latency for 76-byte frame size.

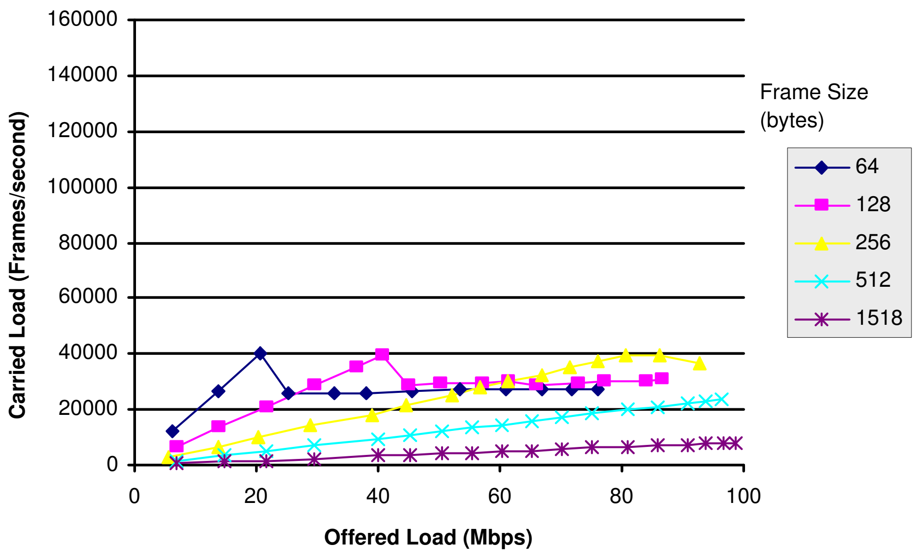


Figure 3-8: Vendor A: Throughput in frames per second with classification and remarking

## 3.3 Vendor A: Jitter

### 3.3.1 Case 1: No classification and no remarking

Figure 3.15 shows maximum jitter versus line utilization with no classification and no remarking. The jitter for all the frame sizes remains close to 150 microseconds, but varies more for higher line utilizations. Figure 3.16 shows frame size on the x-axis and latency on the y-axis, which is portrayed as maximum and minimum to show jitter for 95% line utilization. This graph is for no classification and no remarking. The length of the line shows the maximum jitter and in this case, the lines are comparable for all frame sizes.

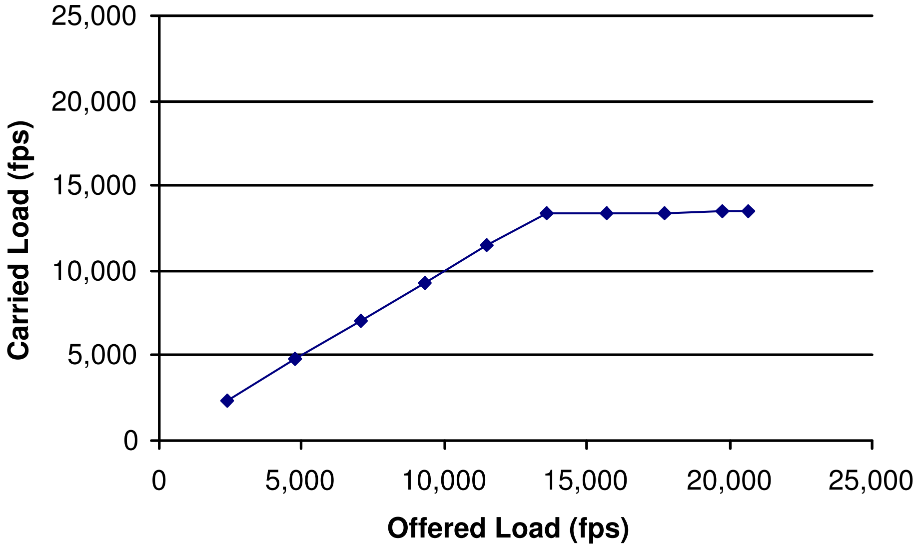


Figure 3-9: Vendor A: Imix traffic with classification and remarking

### 3.3.2 Case 2: Classification and no remarking

Figure 3.17 shows jitter versus line utilization for classification and no remarking. Similar to latency, jitter values show considerable increase only for 76-byte frames for higher line utilization.

Figure 3.18 shows frame size on the x-axis and latency on the y-axis, which is portrayed as maximum and minimum to show jitter for 95% line utilization for classification enabled on the device without remarking. The length of the 76-byte frame line is the longest, showing highest jitter compared to others. Note that the figure has a different y-axis scale than Figure 3.11.

### 3.3.3 Case 3: Classification and remarking

Figure 3.19 shows jitter versus line utilization for classification and remarking. The results are similar to classification and no remarking case, with the jitter increasing

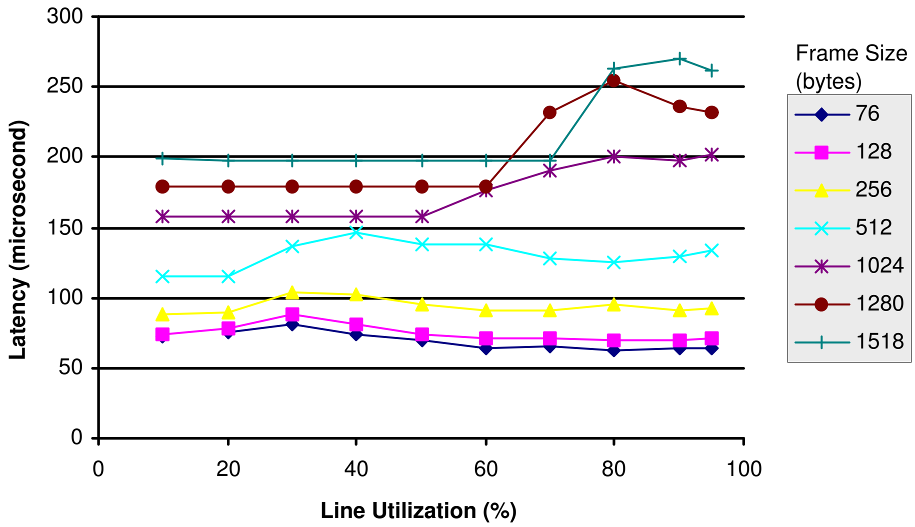


Figure 3-10: Vendor A: Latency with no classification and no remarking

for 76-byte frames. Here the jitter increases earlier as compared to the classification and no remarking case shown in figure 3.17.

Figure 3.20 shows frame size on the x-axis and latency on the y-axis, which is portrayed as maximum and minimum to show jitter for 95% line utilization. This graph is when classification and remarking are configured on the device. The length of the 76-byte frame line is longest showing highest jitter as opposed to others.

## 3.4 Other Vendors

### 3.4.1 Vendor B

This device performed better as compared to vendor A. Throughput for the 512-byte frame size deteriorated for higher bandwidth utilization for all the cases. Performance deteriorated for the 1024-byte frame size for the classification and remarking case for 95% bandwidth utilization. Latency for the 76-byte frame size increased slightly for

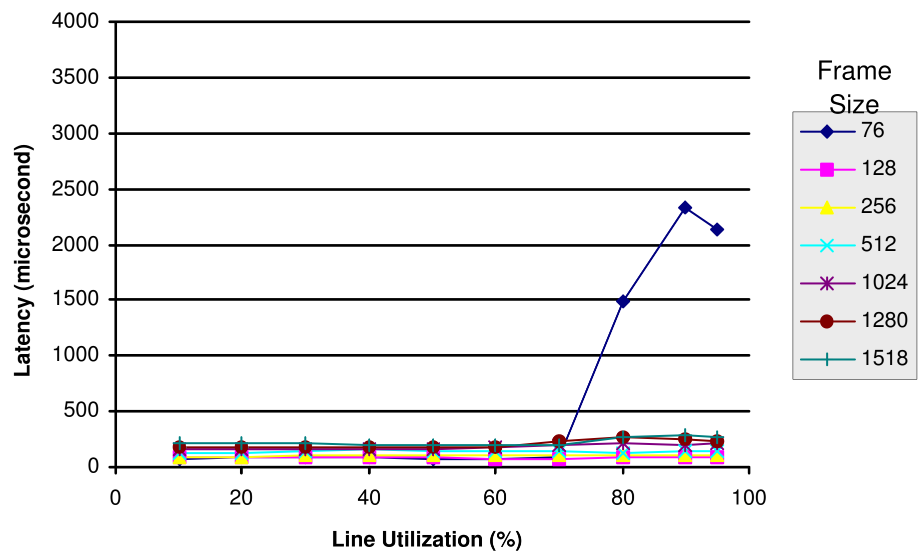


Figure 3-11: Vendor A: Latency with classification and no remarking

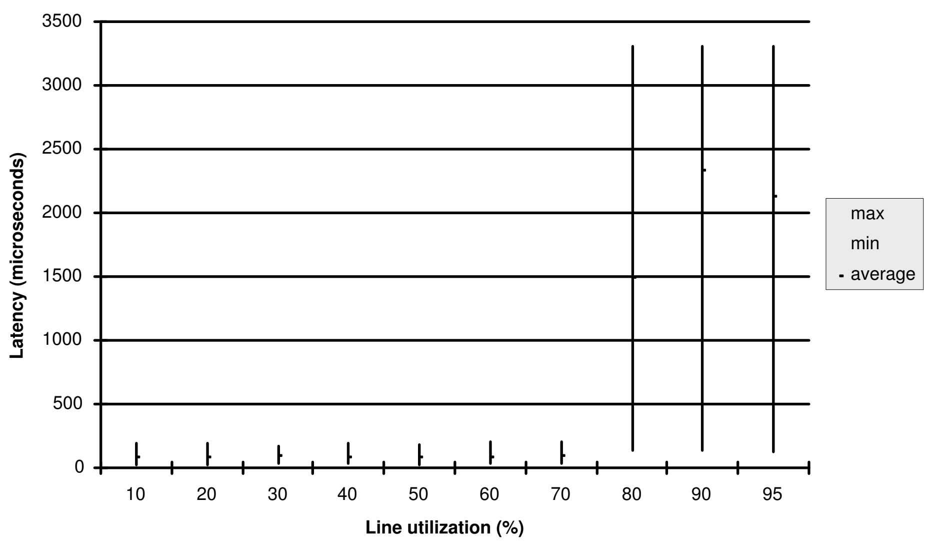


Figure 3-12: Vendor A: Latency for 76-byte frame size showing maximum and minimum with classification and no remarking

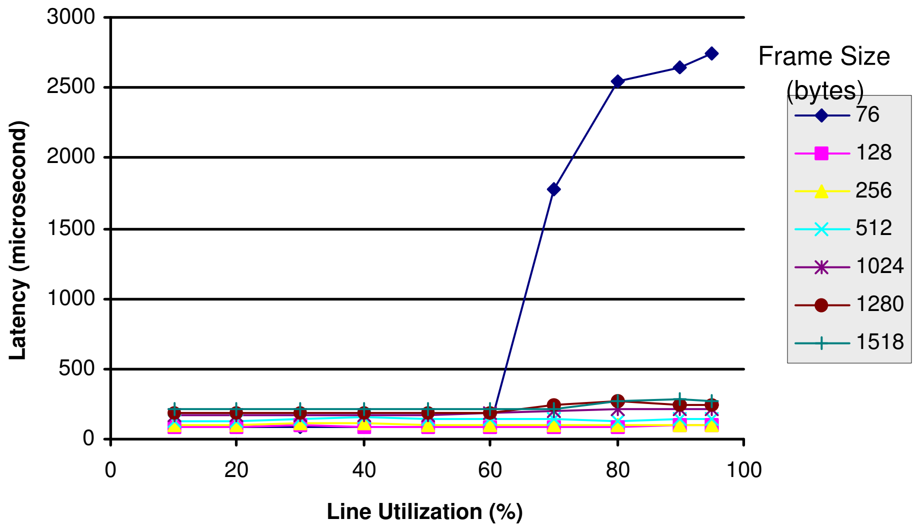


Figure 3-13: Vendor A: Latency with classification and remarking

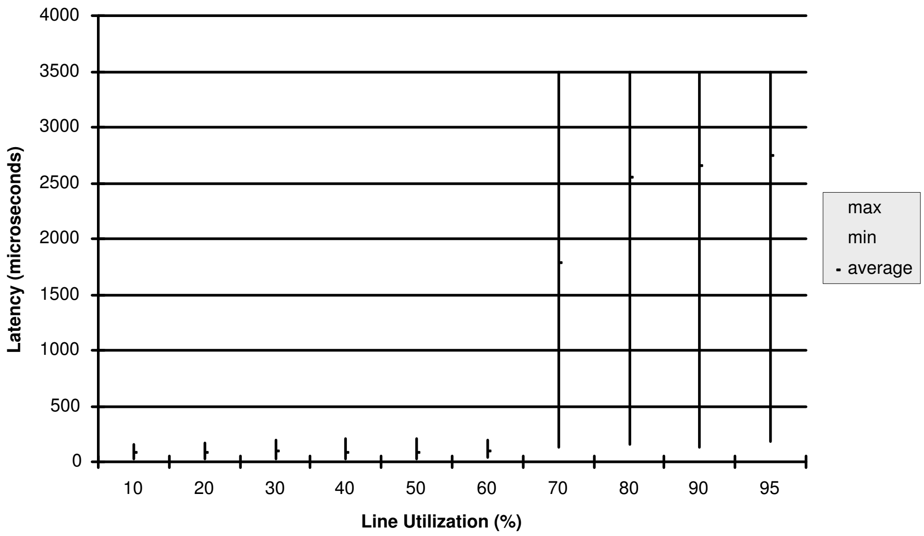


Figure 3-14: Vendor A: Latency for 76-byte frame size showing maximum and minimum with classification and remarking

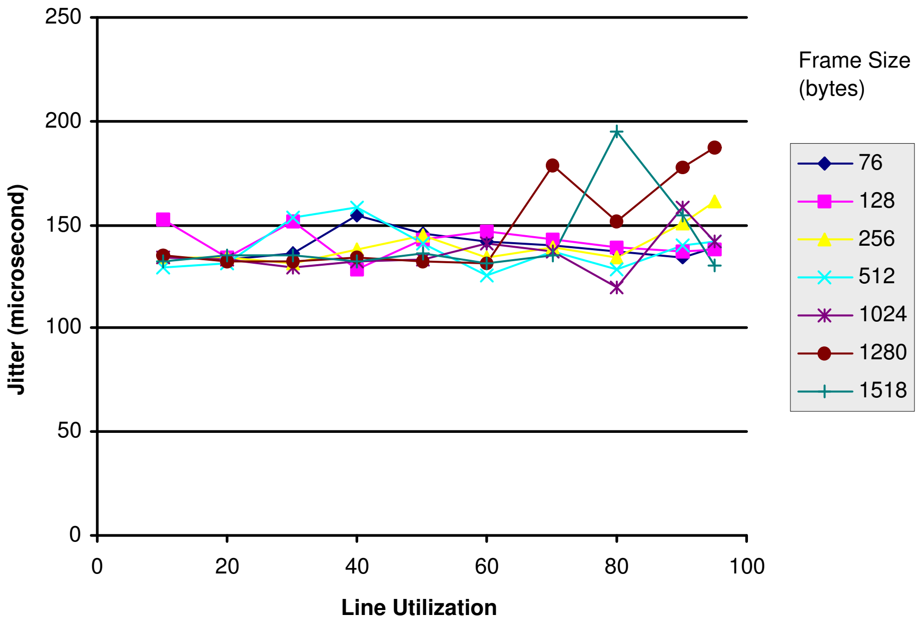


Figure 3-15: Vendor A: Jitter with no classification and no remarking

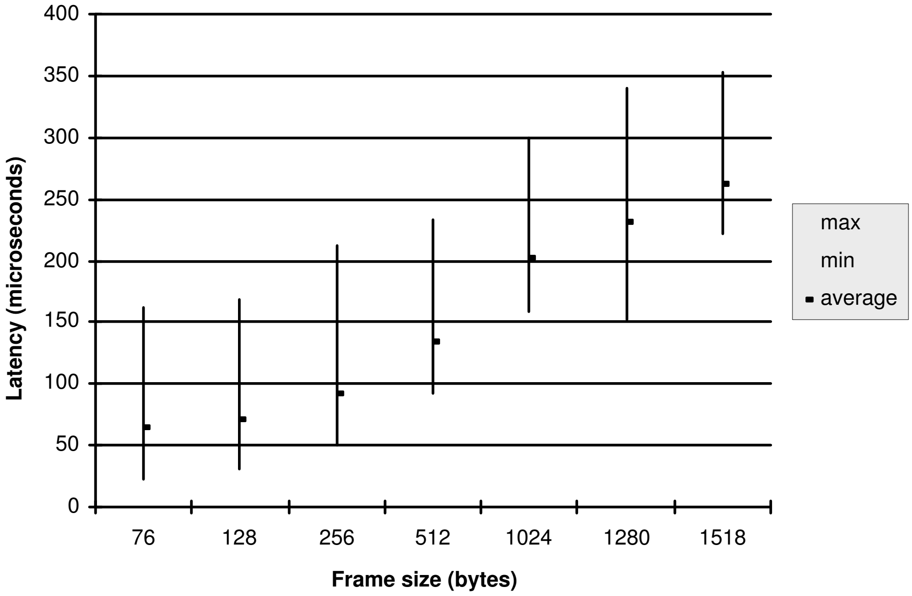


Figure 3-16: Vendor A: High and Low latency for different frame sizes for 95% line rate utilization for no classification and no remarking

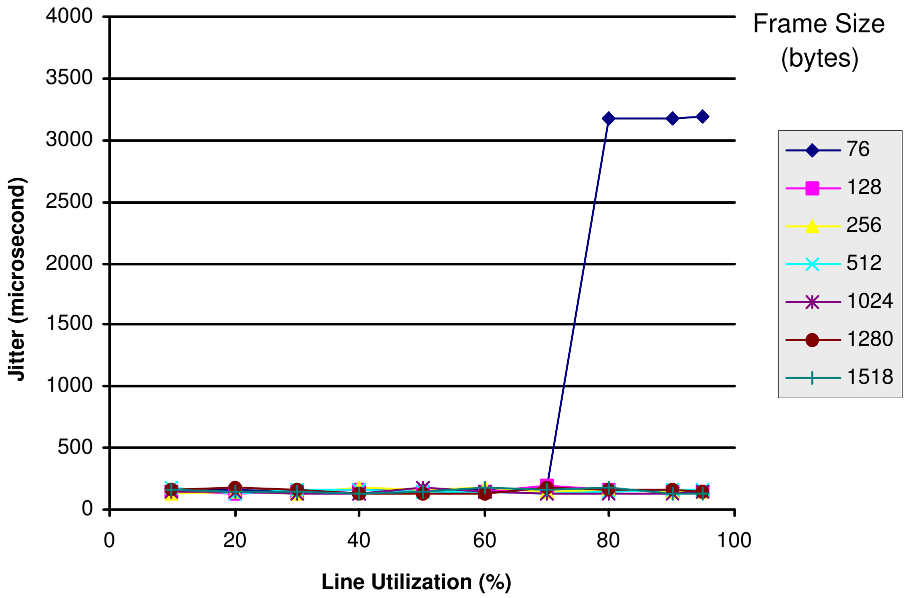


Figure 3-17: Vendor A: Jitter with classification and no remarking

higher bandwidth utilization for all the cases. Jitter was overall higher for the 76-byte frame size as opposed to others for all the cases. Most of the routers have some point in the performance where they behave counter-intuitively.

### Throughput

Figures 3.21, 3.22 and 3.23 show the throughput graphs for case 1, case 2 and case 3 respectively. Throughput graphs does not show much drop in performance as seen in the case of Vendor A. There is a slight drop in performance for 512-byte frame size for higher offered load. Figures 3.24, 3.25 and 3.26 show throughput graphs for Imix traffic for case 1, case 2 and case 3. These graphs do not show any degradation of performance for any case..

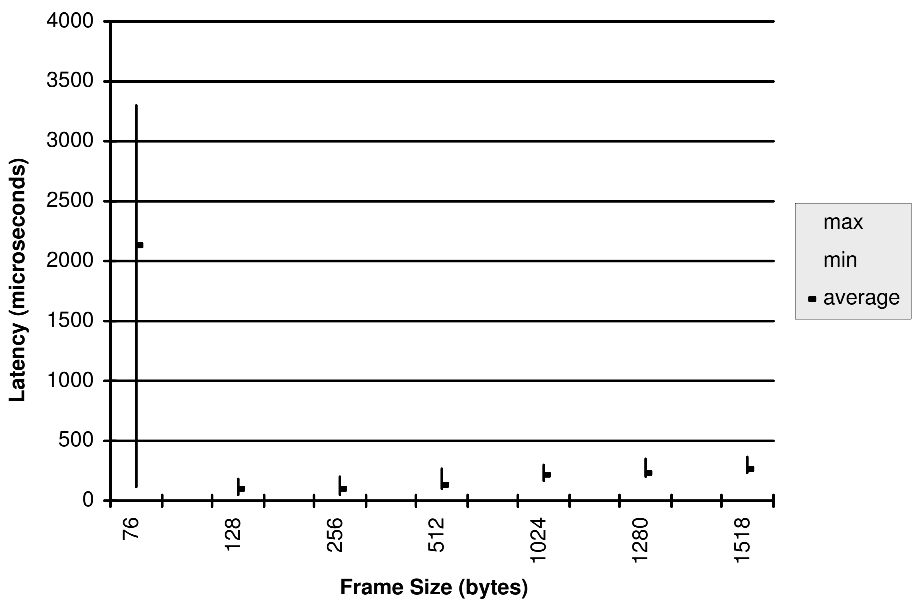


Figure 3-18: Vendor A: High and Low latency for different frame sizes for 95% line rate utilization with classification and no remarking

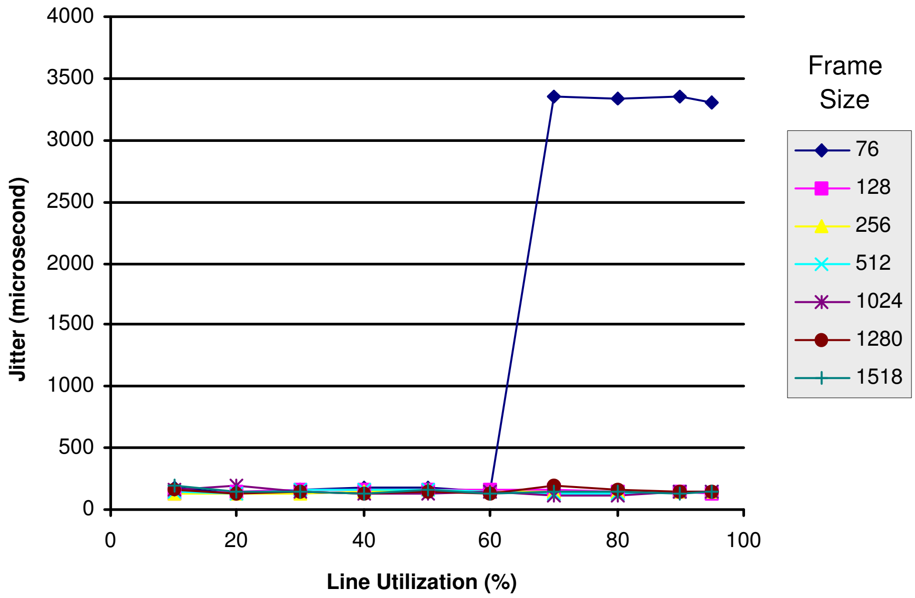


Figure 3-19: Vendor A: Jitter with classification and remarking

### Latency

Figures 3.27, 3.28 and 3.29 show the latency graphs for case 1, case 2 and case 3 respectively. These graph show horizontal lines showing no increase in latency for higher line utilization. As the time required for propagation through the RUT is more for higher frame size, the latency increases for higher frame sizes.

### Jitter

Figures 3.30, 3.31 and 3.32 show the jitter graphs for case 1, case 2 and case 3 respectively. Jitter graphs do not show any sharp rise in jitter. The jitter increases slightly for lower frame size for case 3.

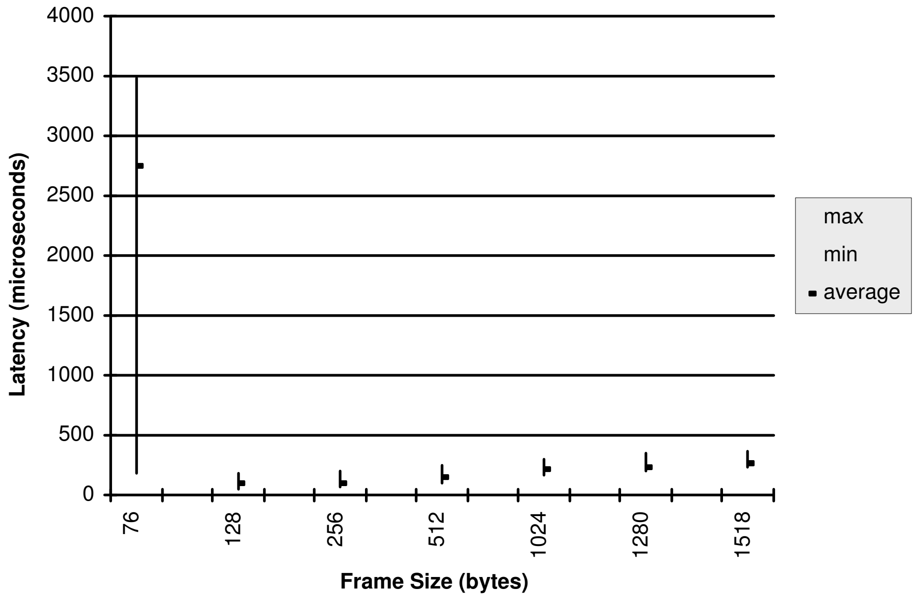


Figure 3-20: Vendor A: High and Low latency for different frame sizes for 95% line rate utilization with classification and remarking

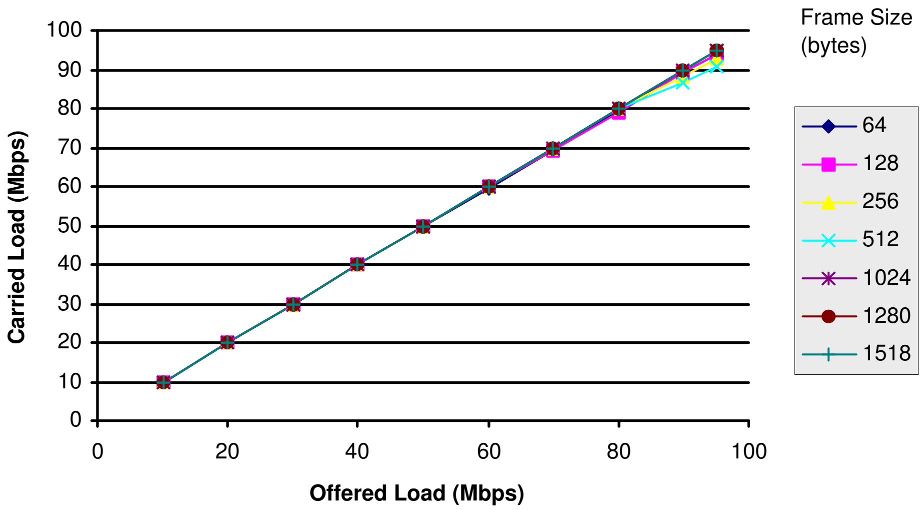


Figure 3-21: Vendor B: Throughput with no classification and no remarking

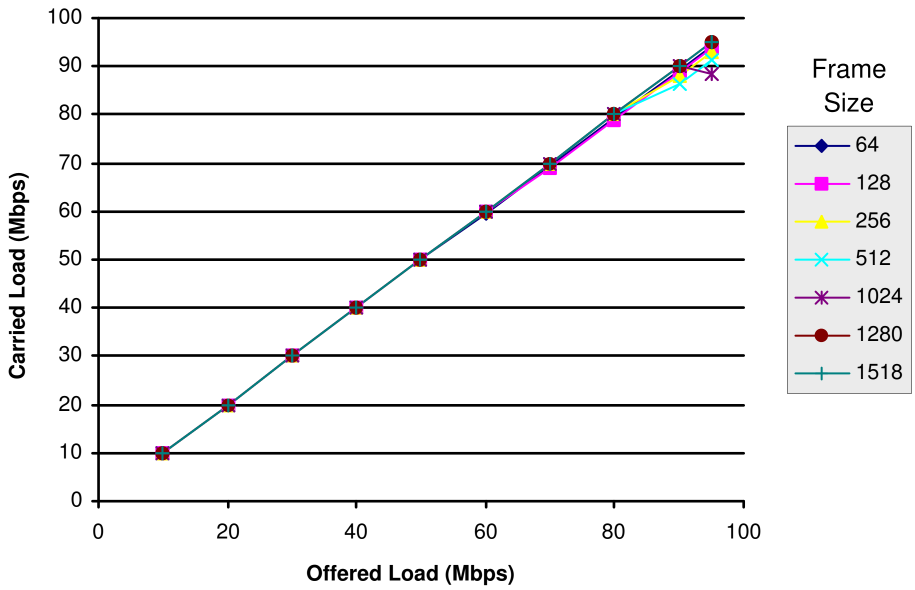


Figure 3-22: Vendor B: Throughput with classification and no remarking

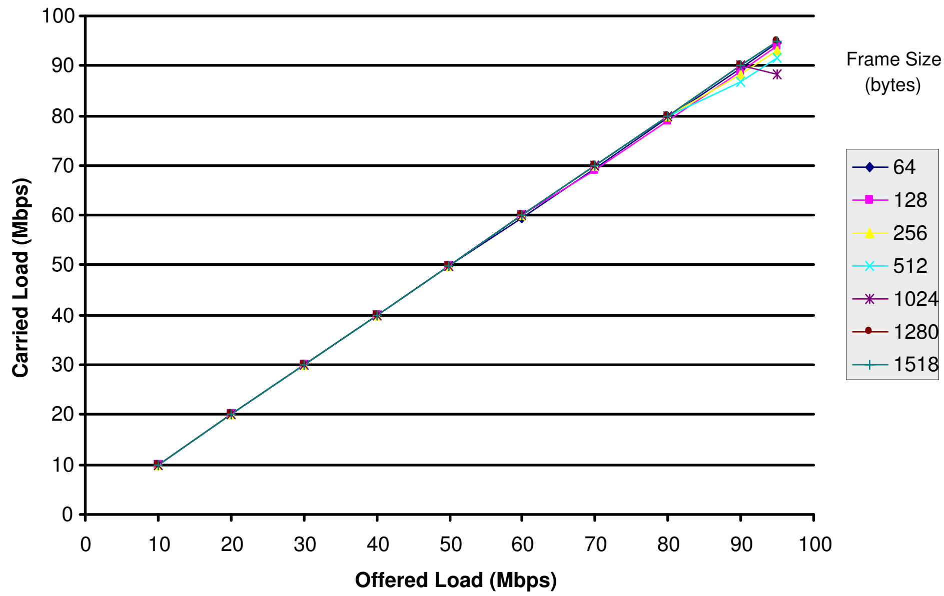


Figure 3-23: Vendor B: Throughput with classification and remarking

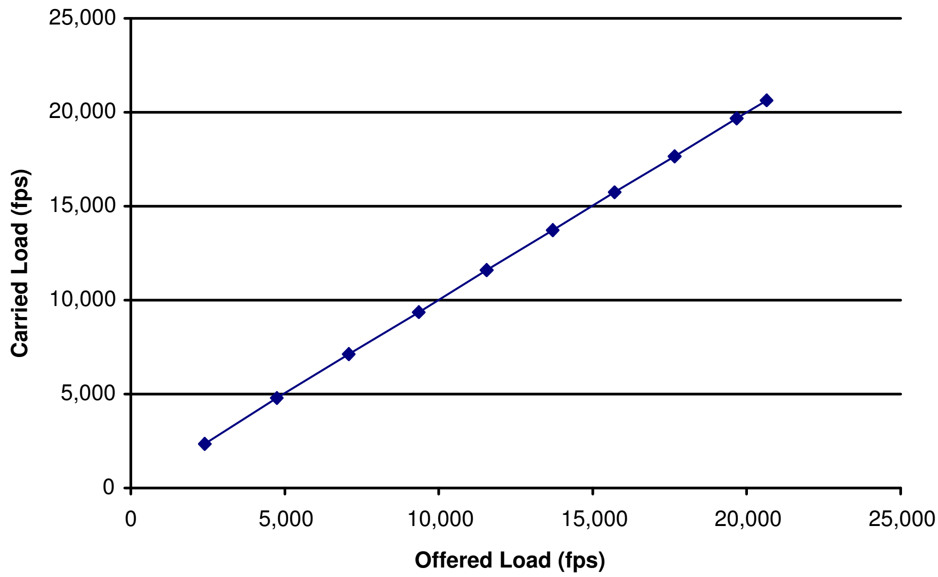


Figure 3-24: Vendor B: Imix traffic with no classification and no remarking

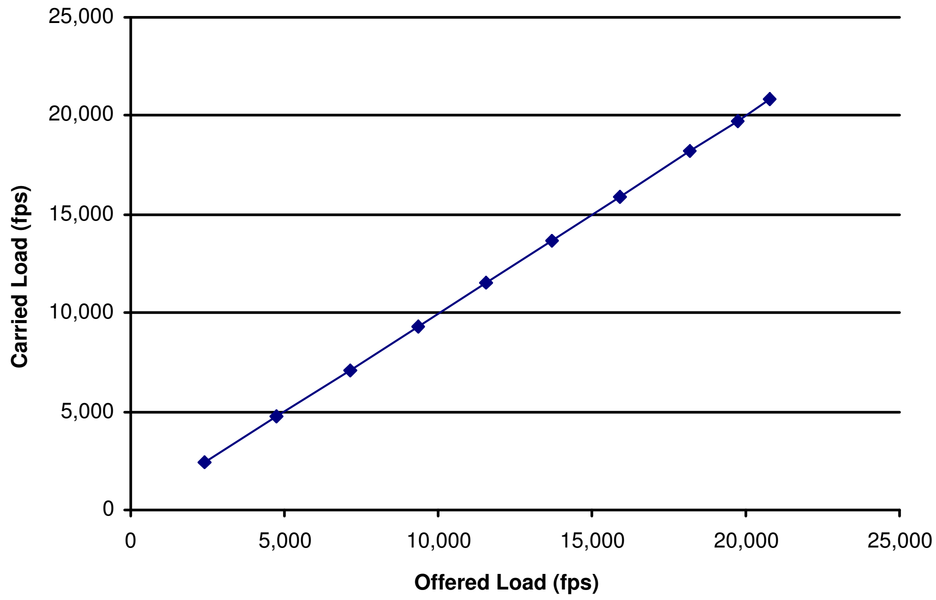


Figure 3-25: Vendor B: Imix traffic with classification and no remarking

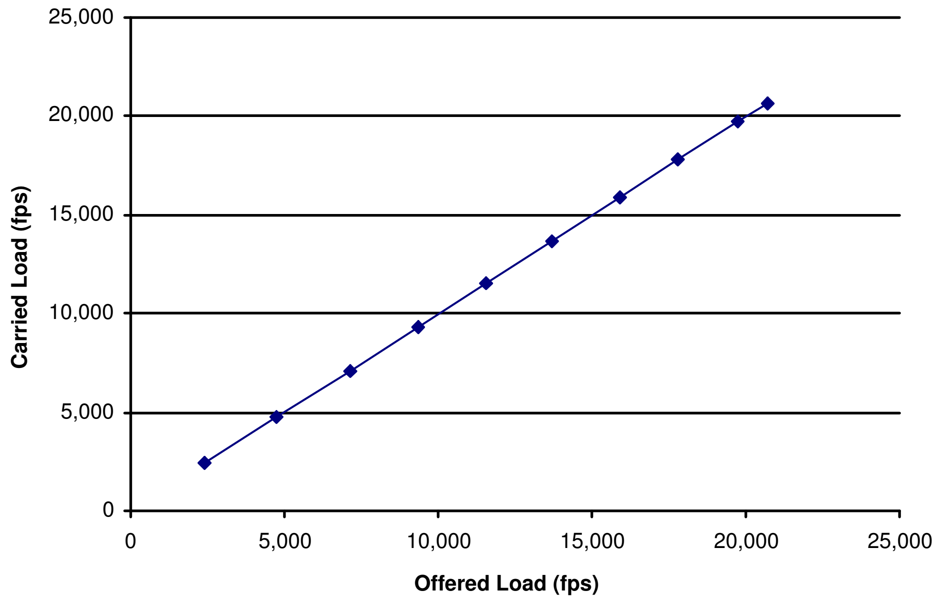


Figure 3-26: Vendor B: Imix traffic with classification and remarking

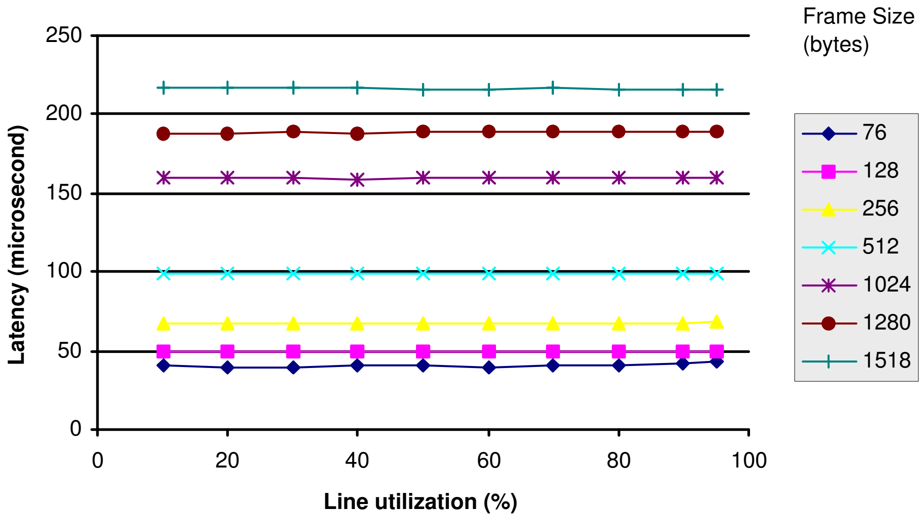


Figure 3-27: Vendor B: Latency with no classification and no remarking

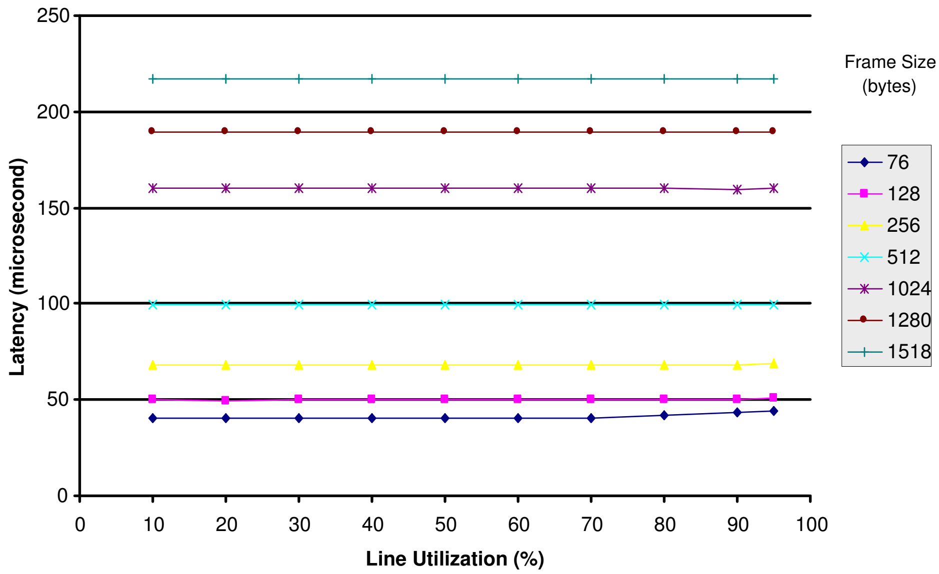


Figure 3-28: Vendor B: Latency with classification and no remarking

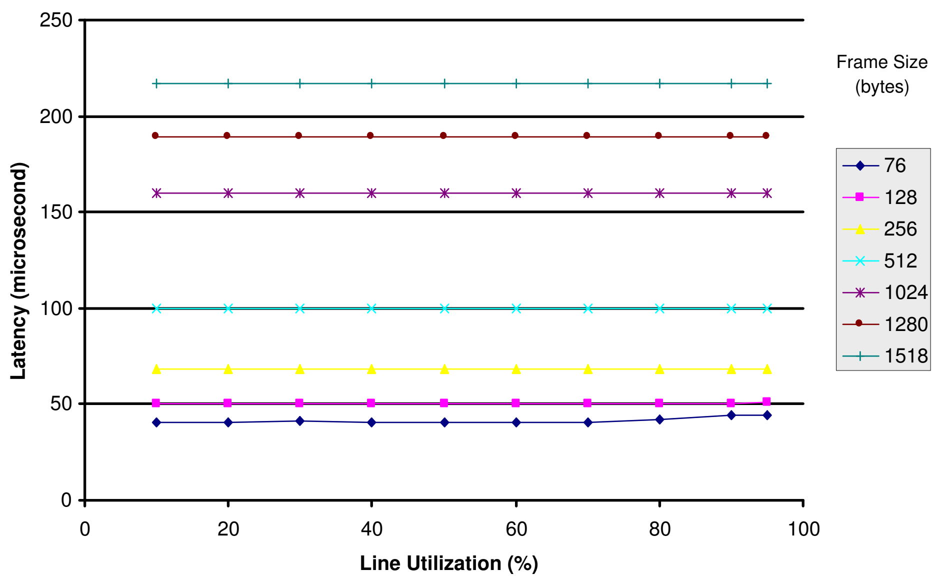


Figure 3-29: Vendor B: Latency with classification and remarking

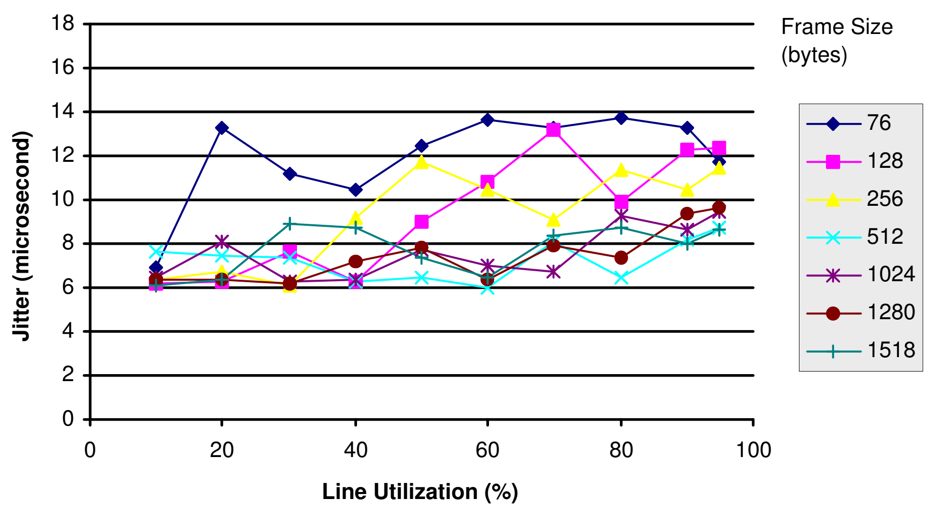


Figure 3-30: Vendor B: Jitter with no classification and no remarking

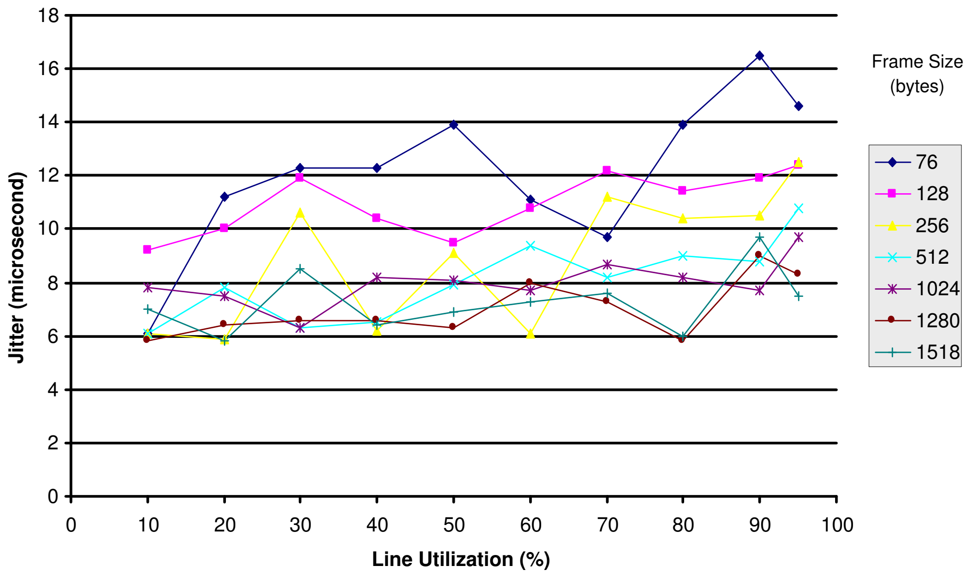


Figure 3-31: Vendor B: Jitter with classification and no remarking

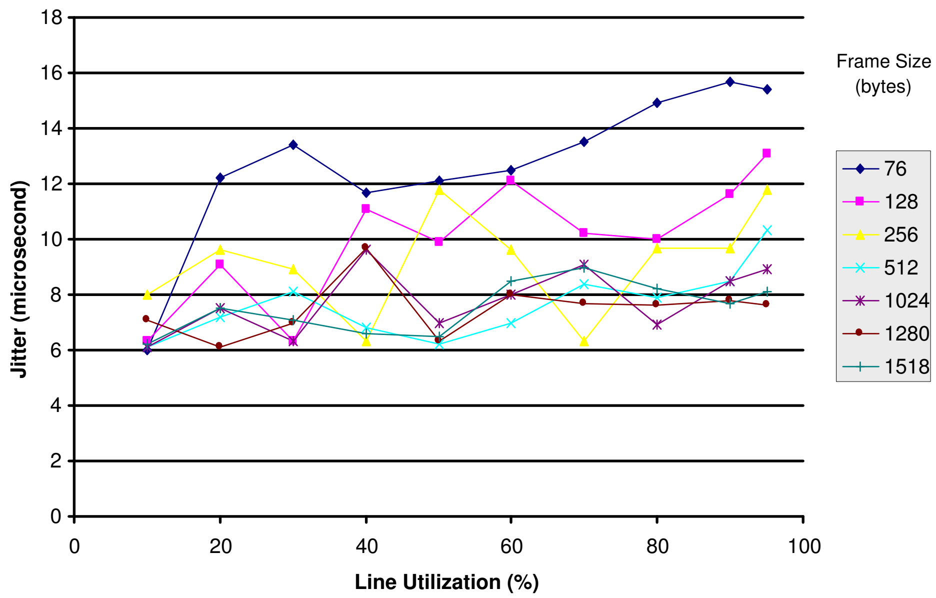


Figure 3-32: Vendor B: Jitter with classification and remarking

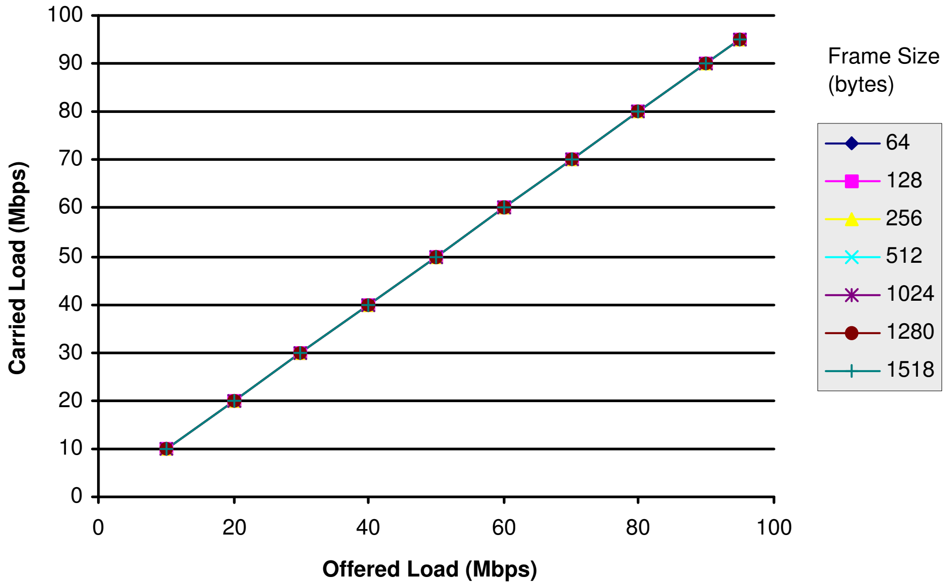


Figure 3-33: Vendor C: Throughput with no classification and no remarking

### 3.4.2 Vendor C

Throughput and latency cases showed expected outcomes with latency higher for larger frame sizes. Jitter increased to large values for higher frame sizes for all the cases. Also high jitter was seen for the no classification and no remarking case with 90% bandwidth utilization.

#### Throughput

Figures 3.33, 3.34 and 3.35 show the throughput graphs for case 1, case 2 and case 3 respectively. The graphs show no drop in performance for any case. Figures 3.36, 3.37 and 3.38 shows throughput graphs for Imix traffic for case 1, case 2 and case 3. For Imix traffic, there is no drop in performance for any case.

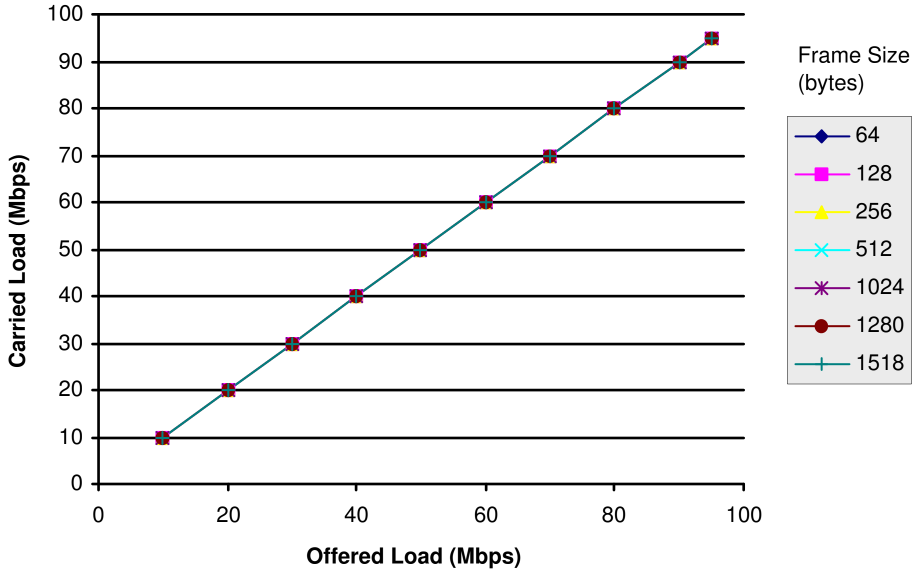


Figure 3-34: Vendor C: Throughput with classification and no remarking

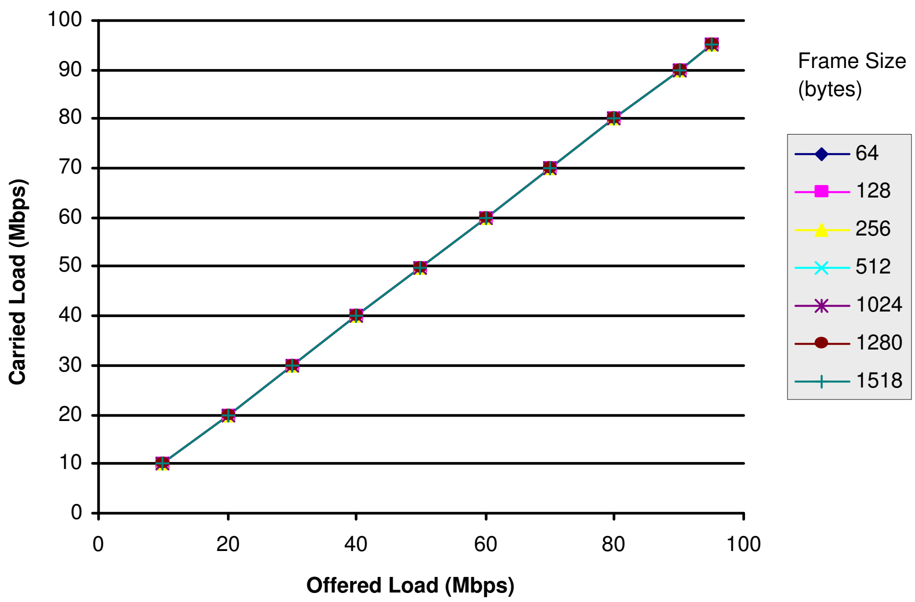


Figure 3-35: Vendor C: Throughput with classification and remarking

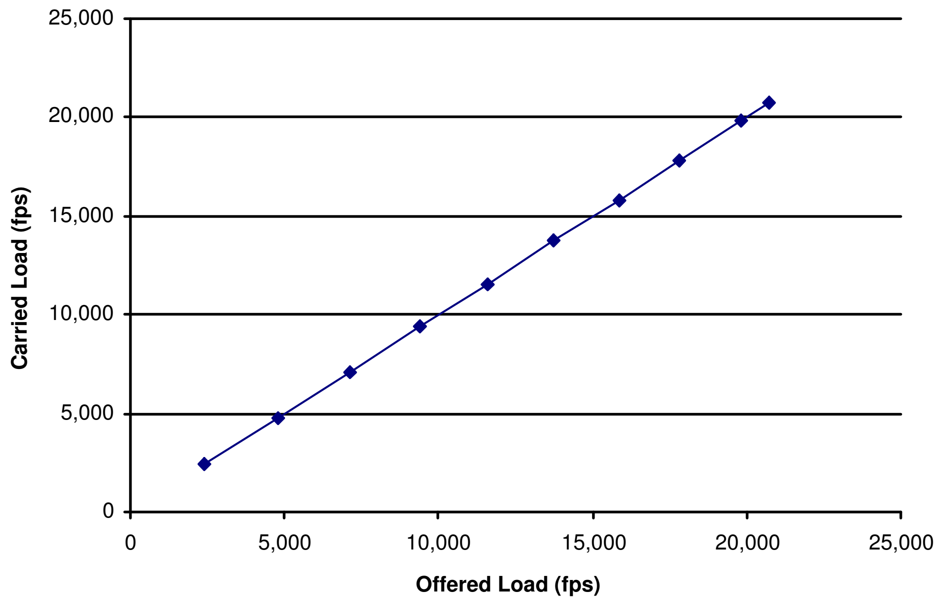


Figure 3-36: Vendor C: Imix traffic with no classification and no remarking

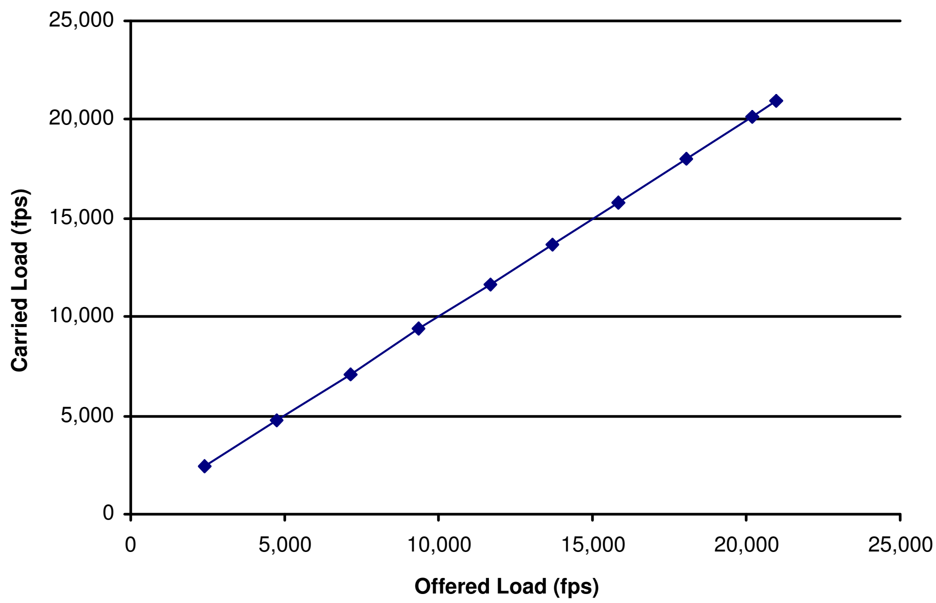


Figure 3-37: Vendor C: Imix traffic with classification and no remarking

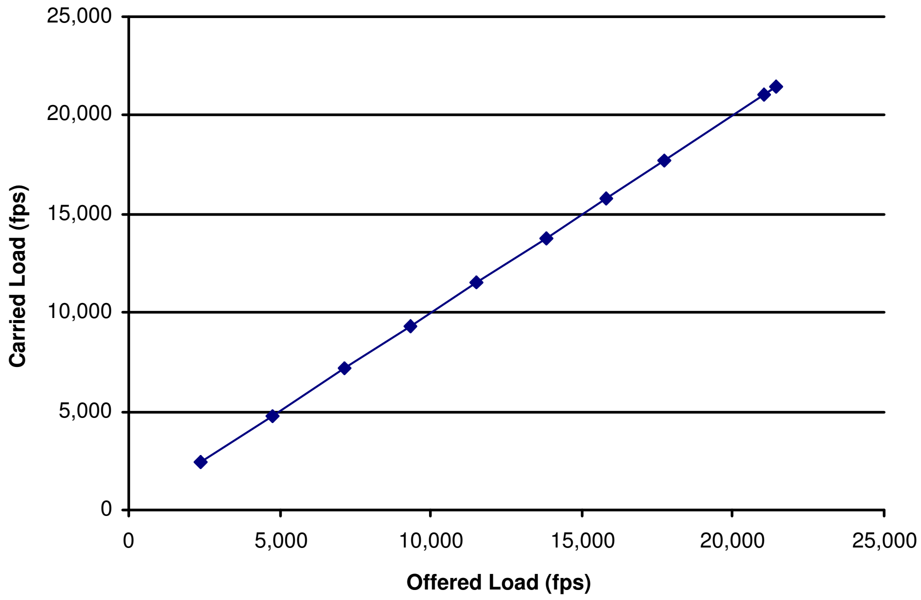


Figure 3-38: Vendor C: Imix traffic with classification and remarking

### Latency

Figures 3.39, 3.40 and 3.41 show the latency graphs for case 1, case 2 and case 3 respectively. These graph show horizontal lines showing no increase in latency for higher line utilization. As the time required for propagation through the RUT is more for higher frame size, the latency increases for higher frame sizes.

### Jitter

Figures 3.42, 3.43 and 3.44 show the jitter graphs for case 1, case 2 and case 3 respectively. The jitter increases for higher line utilization for all the cases. For 256-byte frame size in case 1 there is an unusual jump in the jitter.

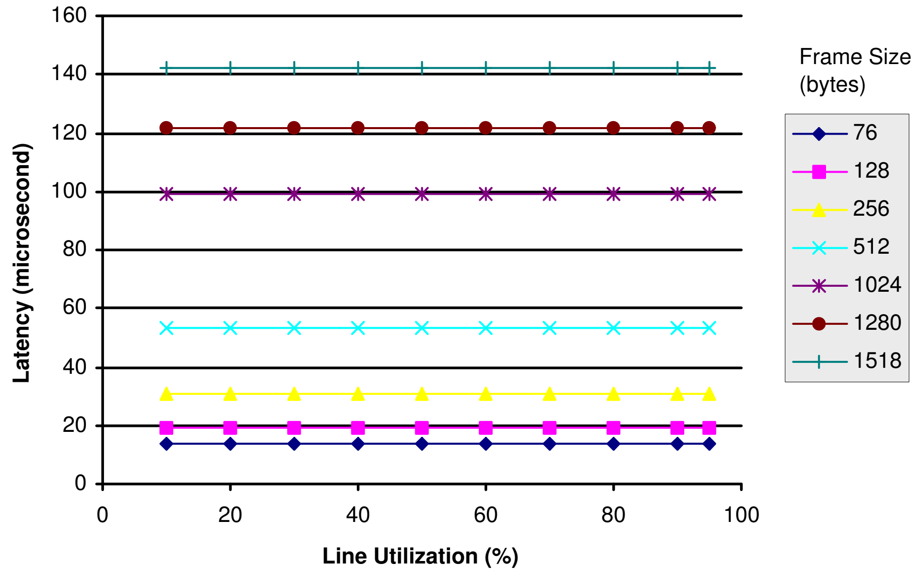


Figure 3-39: Vendor C: Latency with no classification and no remarking

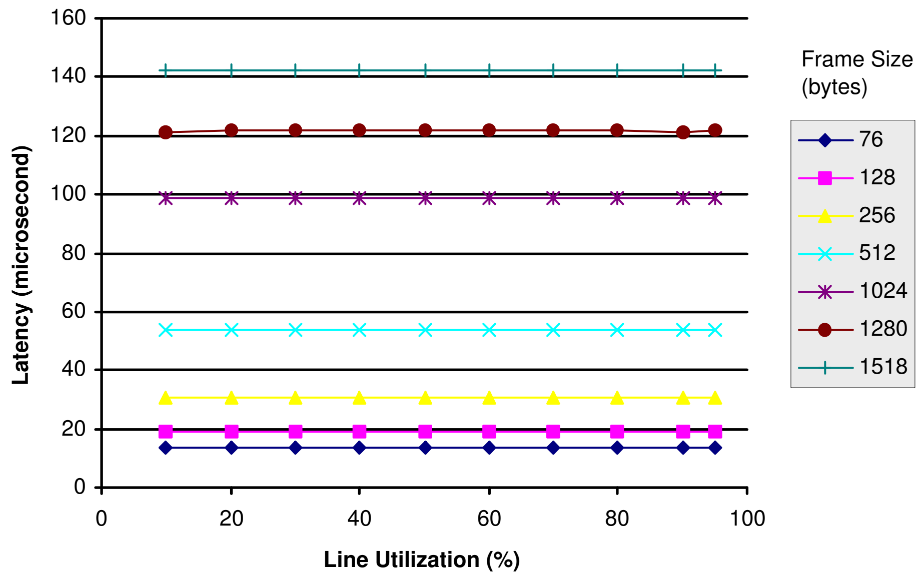


Figure 3-40: Vendor C: Latency with classification and no remarking

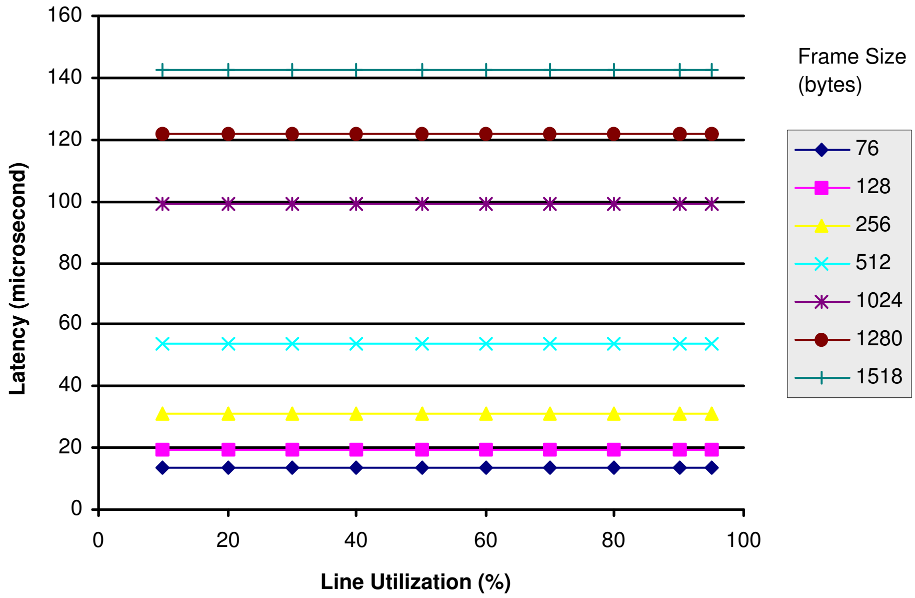


Figure 3-41: Vendor C: Latency with classification and remarking

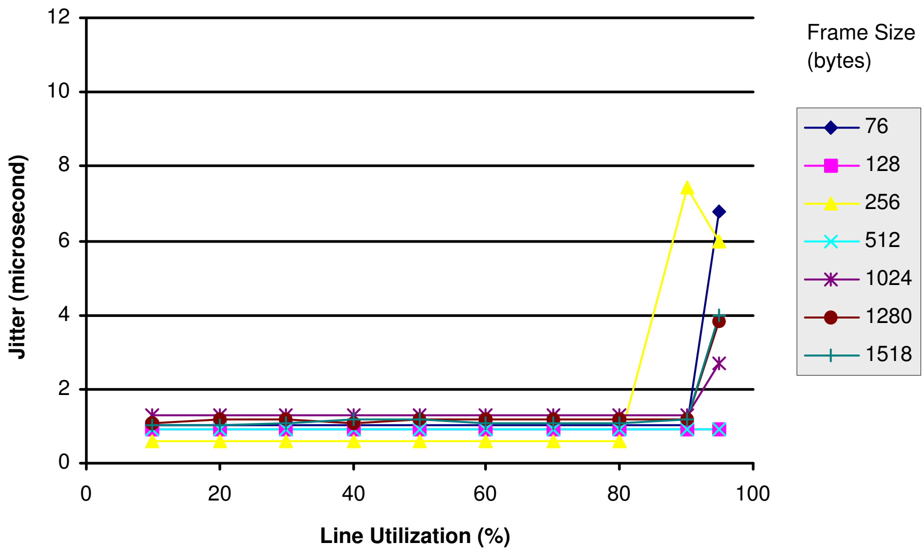


Figure 3-42: Vendor C: Jitter with no classification and no remarking

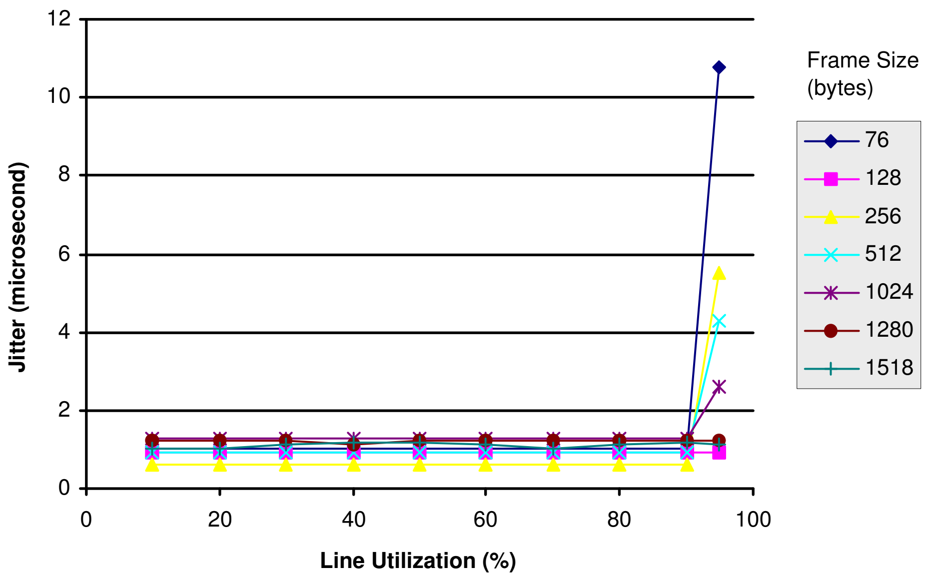


Figure 3-43: Vendor C: Jitter with classification and no remarking

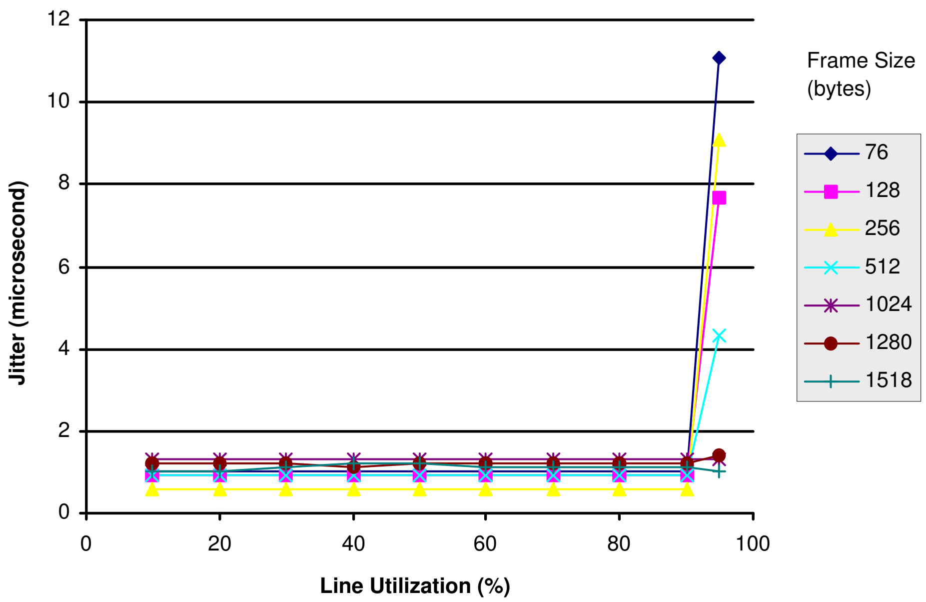


Figure 3-44: Vendor C: Jitter with classification and remarking

### 3.4.3 Vendor D

Vendor D device was subjected to the same tests as other devices, but as soon as classification and/or remarking were enabled, performance of the device started deteriorating and it rebooted repeatedly. This behavior showed that the device was incapable of handling QoS aspects like classification and remarking. This device was noted to be older than vendor B and C.

## 3.5 Conclusion

The following conclusions were made based on the experiments.

### 3.5.1 Classification and remarking deteriorates performance

Normally when a packet passes through a router it goes through a highly optimized forwarding technique, typically used for processing of the packet. If classification is enabled based on source IP, it forces the router to look into the packet header source IP, for matching. This proves expensive on the resources of the router as it has to do non standard function of classification.

Similar is the case when classification and remarking are enabled. After classification, the router updates the ToS field in IP header. This requires more resources on a per packet basis and hence taxes the router more.

### 3.5.2 Performance differences

We have observed performance differences that vary widely from one vendor to another. The reasons that were found to explain possible differences in the behavior are:

### Age of the device

The older devices from an architectural point of view showed inferior performance. This could be because designers of the newer devices were better in predicting the computational requirements a device needs as opposed to the foresight in the design of older devices.

### Software versus hardware forwarding

Software based forwarding is slower than hardware based [11]. There are implementations which rely on the computational speed of the CPU for certain type of forwarding. If the traffic forwarding falls into the category such that it needs software based forwarding, then the performance will be affected.

As most of the implementation details are proprietary, it cannot be confirmed which vendor used what techniques.

### Soft spots

Every once in a while we see a device behave differently which is not consistent with the behavior for rest of the testing outcomes. Vendor B showed slight performance degradation for the 512-byte frame size for higher offered load. Vendor C showed increased jitter for 90% bandwidth utilization for no classification and no remarking as opposed to other cases. We called these specific conditions under which the device behaves counter-intuitively as ”soft spots” of the device.

### Imix traffic

Imix traffic does not make the performance of the device change dramatically. As we have seen, that the devices' performance were consistent with the single packet size test outcome. For vendor A, there was performance degradation for classified traffic as well as classified and remarked traffic. For vendor B, there was no degradation of performance for any case. For vendor C also, there was no degradation in the performance for any case.

# Chapter 4

# Summary and Future Work

This chapter discusses the thesis summary and future work.

## 4.1 Summary

Quality of Service testing methodology should effectively test an IP network implementation. Testing multiple implementations helps us refine the methodology and also helps in identifying problem areas in an implementation.

For developing the methodology we first identified the parameters that are important and can be changed on the device. Second, the metrics to be measured were identified. One of the goals of the project was to develop methods that take into account the constraints of the available testing tools. Following the initial results, more relevant test cases were identified and developed. Different methods for measuring jitter were also discussed. A study of different methods with typical traffic cases was done to show their advantages and disadvantages of each method.

Preliminary experiments were performed on individual devices. Throughput, latency, and jitter are the metrics commonly identified to be relevant from the QoS point of view. Throughput for vendor A deteriorated when classification and remarking were enabled. Looking at the throughput graphs in frames per second showed the device performance was limited to lower frames per second for classification and remarking. Vendor A performed in similar fashion for Imix traffic as well. Latency and jitter also increased for classification and remarking. For vendor B, there was no major deterioration in performance with classification and remarking. There were some soft spots in the performance as noted in Chapter 3. Vendor C showed no deterioration in performance for throughput for any of the cases. The latency graph also showed expected behavior, but jitter increased for high load conditions. Vendor D could not handle classification and remarking. The device rebooted repeatedly while the experiments were performed and was not considered in the results.

The experiments were performed on devices which are currently deployed in many ISP's network edge. Assuming a mixture of different vendor devices being deployed along a path of a communication, the results show that it is not easy to assure the customer some QoS, as all the devices tested do not perform well under high load conditions on all metrics. So to assure a level of service, an ISP will have to make sure that all the devices along the path are capable of handling the required tasks without any performance degradation. This might call for replacing some of the existing devices in the network.

## 4.2 Future Work

This thesis can be enhanced in a few ways. Below we discuss the future work based on the study, experiments and the results observed.

### 4.2.1 Multiple priority traffic

The tests performed for this thesis just have single priority traffic. It is interesting to compare how a device behaves when it is subjected to different priority traffic streams. This will also tell more about the effectiveness of the queuing schemes being used in the device. The outcomes could be looked at from both conformance and performance aspects.

### 4.2.2 Study of other aspects of QoS

Quality of Service aspects like queuing and congestion management could be considered dedicated study topics. Different congestion management techniques could be compared. Different implementations of congestion management could be compared with the help of multiple vendor devices. It is also important to know the role queuing plays in the congestion management solution.

### 4.2.3 QoS applied to other environments

The tests done for this thesis involve IP network. Looking into other environments like MPLS or wireless is also important from the end-to-end QoS deployment perspective.

MPLS has always been marketed as a better QoS environment, and studying that for performance could be very interesting.

Quality of Service when provided over a wireless medium has a different set of challenges. 802.11e specifies the QoS aspects of a wireless network. Performance of an 802.11e implementation could lead to interesting findings.

### 4.2.4 Development of generator/analyzer to measure line rate per packet latency and other methods of jitter

It would be very helpful to develop a per packet latency measuring tool that can operate at a line rate of at least 100Mbps. As the guarantees given to the customers and enterprise customers do not involve measuring per packet latency, only the average, high and low latency measurements are provided by test equipment vendors. Service providers currently do not demand such a tool, however measuring per packet latency is very important from a study, research and network planning perspective. This will enable us to measure the other methods of jitter discussed in Chapter 2.

### 4.2.5 Layer 2 QoS

IEEE 802.1Q specifies the Layer 2 priority tagging mechanism in the VLAN tag. This is popularly referred to as 802.1p and has recently been incorporated into the 802.1Q standard. This thesis could be extended in the future to incorporate scenarios that include Layer 2 Ethernet Bridges (or switches).

### 4.2.6 Application QoS testing

Application layer testing involving QoS measurements for voice, video and data traffic is interesting from the end user perspective. Layer 4-7 testers could be utilized for this testing.

# Bibliography


[1] Geoff Huston, "Internet Performance Survival Guide QoS strategies for Multiservice Networks", *Wiley Computer Publishings*, 2000.

[2] Zheng Wang, "Internet QoS Architectures and Mechanisms for Quality or Service", *Morgan Kaufmann Publishers*, 2001.

[3] S. Bradner and J. McQuaid, "Benchmarking Methodolgy for Network Interconnect Devices", *IETF RFC 2544*, March 1999.

[4] William Stallings, "Data and Computer Communications", *Prentice Hall*, 2000.

[5] Harry Newton, "Newtons Telecom Dictionary", *CMP Books*, February 2001.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architechture for Differentiated Services", *IETF RFC 2475*, December 1998.

[7] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services field (DS Field) in the IPv4 and IPv6 Headers", *IETF RFC 2474*, December 1998.

[8] M. Bustos, T. Van Herck and M. Kaeo, "Terminology for Benchmarking IPSec Devices", *IETF draft-ietf-bmwg-ipsec-term-02.txt*, November 2003.

[9] Foundry Networks, "White Paper: Voice over IP", *http://www.foundrynet.com/solutions/appNotes/VoIP.html*, 2004.

[10] Jerry Perser et al., "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", *IETF Draft, draft-ietf-bmwg-dsmterm-00.txt*, February 2001.

[11] Cisco Systems Inc., "Configuring Cisco Express Forwarding", *http://www.cisco.com/en/US/products/hw/switches/ps4324/ products_configuration_guide_chapter09186a00802c3051.html#wp1142909*, 2004.

[12] Sean McCreary, "Packet Length Distributions", *http://www.caida.org/analysis/AIX/plen_hist*, 2002.

[13] Light Reading, Inc., "Modeling Tomorrows Internet", *http://www.lightreading.com/document.asp?doc_id=25457&sitelightreading*, 2002.

[14] Applied Global Technologies, "Jitter Test Algorithms and Program For H.323 Video Conferencing", *http://www.appliedglobal.com/files/whitepapers/WP_Jitter %20Test%20Algorithms%20And%20Program111.doc*, 1999.

[15] Ahsan Habib, Sonia Fahmy and Bharat Bhargava, "On Monitoring and Controlling of QoS Netwrok Domains", *CERIAS and the Department of Computer Science, Purdue University*, 2002.