University of Cyprus
Computer Science

Implementation and Evaluation of Differentiated Services on Linux

by
Yiannos Mylonas

Supervisor
Dr. Andreas Pitsillides

A thesis submitted to the Graduate Faculty of University of Cyprus in partial fulfillment of the requirements for the Degree of Master of Science at Computer Science department.

To My Fiancé: Dora

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr. Andreas Pitsillides for his support and encouragement throughout the duration of the project. I would like also to thank Dr. Costas Pattichis for his support and help for accomplishing my goals. I would like to thank Marino Stylianou and Melino Kyriakou for helping me setting up the Linux network and for all their patience at rough times.

Mere words are insufficient to express my gratitude to my parents, for their love and incredible support, without which this would not have been possible.

Abstract

The existing Internet architecture is based on the “best effort” model for delivering packets across the Internet. The current architecture delivers a packet at its best possible (best-effort) but doesn’t guarantee when it will be delivered. Nowadays, the users work and play habits are changing, e.g. users expect to watch movies through the network, play 3-D games, and check their stock online, videoconference and other. The demands of the users have changed dramatically since the creation of IP, where it was mostly used for email and ftp. Another new application is the WWW that has been widely used worldwide. WWW has created a new friendly interface for the user, and stimulated further demands from the network.

The existing architecture of IP is inadequate to handle new applications. Time critical applications such as video, audio and several others have created an even greater demand on the Internet. Recently, several different solutions were proposed, but most have failed to replace IP.
Lately, several new protocols and architecture were proposed to enable basic quality of service provision in Internet. In this thesis we investigate the Differentiated Services (DiffServ) architecture. DiffServ is a new architecture based on the concept of aggregated differentiated treatment of services. DiffServ was proposed in 1997. Since then it has attracted a lot of attention by many researchers.

The aim of this thesis is to implement a differentiated services pilot network in Linux environment and investigate the performance of various network functions, that may provide differentiated quality of service. These functions include various queuing disciplines such as pFifo, RED, and TBF. Through the pilot network we aim to investigate different ways to implement differentiated networks and present recommendations for different network traffic and conditions.

Table of Contents

Table of Figures

1 Introduction

1.1 IP Quality of Service

1.1.1 Introduction to IP QOS

The existing Internet architecture is based on the “best effort” model for delivering packets across the Internet. The current architecture delivers a packet at its best possible (best-effort) but doesn’t guarantee when it will be delivered. The IP has succeeded in meeting the requirements of its designers at the time it was implemented. At that time the expectations of the users' were very low, in terms of the variety of services and the quality of service offered to them. However, nowadays IP can't scale very well with increasing demands by the users in terms of supporting a variety of increasingly integrated services, with more predictable quality. The users work and play habits are changing, e.g. users expect to watch movies through the network, play 3-D games, check their stock online, videoconference and other. The demands of the users have changed dramatically since the creation of IP, where it was mostly used for email, ftp, and lately the World Wide Web (WWW, or the web). The WWW has created a new friendly interface for the user, and has been widely adopted (some suggesting that it is the main reason for the phenomenal adoption of the Internet). It has stimulated new demands and requirements for the computer networks.

The existing architecture of IP is inadequate to handle new applications. Time critical applications such as video, audio and several others have created an even greater demand (in terms of expected quality of service provision) on the Internet. Lately, several alternative solutions were proposed, but most have failed to replace IP.

One of these proposals and the most threatening to the IP architecture is the ATM architecture. One may argue that ATM has succeeded to win the technical battle for the provision of (Quality of Service) QoS to the users (i.e. better service provision, in comparison to the IP), but lost the battle in the applications domain. Not many applications that run under pure ATM can be identified.

The ATM is a very expensive (in terms of bandwidth and efficiency) protocol to have and without the pure ATM applications there is not a lot to gain. The IP has the advantage of many well-established applications, and because of its simplicity, it offers a more cost effective solution but not with inbuilt service guarantee in its present form. ATM is currently used for backbone but it does not appear that it will win the battle to the doorstep. In order to make IP better able to support some form of Quality of Service provision to the users, several new architectures are proposed.

Quality of Services as seen by the customer is affected by the performance of several layers of the TCP/IP stack, including the application and network related functions. In this thesis we will focus on the effect of the network on the delivered QoS.

1.1.2 QoS Definition

The main target of the QoS is to satisfy customers' needs. The word QoS has different meanings among people. Even though, there are different views on the definition of the QoS, there is an agreement on the key concepts and on the terminology of QoS. Class of Service is a more general term that is used to describe a set of features and other characteristics available with a specific service. A QoS service is a term used to specify a set of performance characteristics for a service. Some of those characteristics are: service availability, delay and delay variation, throughput and packet loss rate [13].

The QoS is always limited by the weakest link in the chain along the path, between the sender and receiver. The most critical characteristics of QoS are minimizing delivery delay, minimizing delay variations and providing consistent data throughput capacity. These QoS characteristics should be provided together with efficient use of the limited bandwidth resources. The ideal performance, from a link viewpoint, is to be able to use the link bandwidth efficiently. The most critical characteristics of QoS are:

Minimizing delivery delay

Minimizing delay variations

Providing consistent data throughput capacity

Minimizing Losses

1.1.3 Parameters of QoS

To be able to implement a QoS certain parameters need to be defined by the applications. These parameters will help us to implement QoS for our customers. Some of these parameters are the following:

Latency
Jitter
Bandwidth
Packet Loss
Availability.

QoS Terminology

Classes

Classes, this term is used to categorize the users or applications in different classes, such as Premium, Assured and Best-effort. Classes will be discussed in more detail later on.

Latency

Latency is referred to as the time it takes to send a message from the sender until to the time it is received by the receiver (i.e. end-to-end delay experienced by a packet).

Router Latency

It’s the time it takes a router to retransmit the packet once it has arrived at the router.

Jitter (Delay variation)

Refers to the variation in time delay between all packets in a session. This parameter can be critical, as for example when sending a video stream over the network and the packets arrive with a big variation in the delay between them. This affects the quality of the playback, and if the variation in delay is very high it can distort our video to unacceptable levels.

Bandwidth

Bandwidth is the ideal capacity that the network can operate. The networks never work on ideal maximum capacity since there are negative factors that cause deterioration of the quality of the network. Such factors include transmission delay, noise, etc.

Packet Loss

Packet loss takes place when we are experiencing congestion on our network. This parameter is the maximum packet loss we can accept. In the event of network congestion this parameter may be used to discard packets intelligently, up the defined Packet Loss parameter.

Service Availability

Availability is the reliability of the user's connection to the Internet service. In other words what is the probability to connect to my service provider network when I want to. In order to be able to maintain all these parameter there is a need of Service Level Agreement (SLA).

1.1.4 Service Level Agreement

SLA is a contract between the service provider and the customer. The SLA can be applied to a customer, a group of customers, or a group of businesses. The SLA defines end-to-end service specifications and may consist of the following:

Availability
Services offered
Service Guarantees
Responsibilities
Auditing the service
Pricing

Availability-guarantee uptime, service latency. It's the time it takes for the user to access the network.

Services offered-the specification of the service levels offered.

Service Guarantees-for each class. The service guarantees are the guarantees for the throughput, loss rate, delay, delay variation and class over-subscription handling for each class. For instance if the premium class and best effort get the same guarantees then there is no reason for paying more money to belong in the premium class.

Responsibilities-In case the ISP breaks the SLA what the consequences are. Does the ISP have 24 hours support?

Auditing the service-Does the ISP or the customer have the software or the tools to audit the connections?

Pricing-It's a very hot topic under discussion and research that addresses the issue of pricing according the SLA that the client had requested.

The Service Level Specifications and /or Service Level Objectives (SLOs) describe in more detail the characteristics of the SLA.

The Service Level Specification, SLS, consists of following:

- *Expected throughput*
- *Drop probability*
- *Latency*
- *Constraints on the ingress*
- *Constraints on the egress points*

- *Scope of service*
- *Traffic profiles*

An SLO partitions an SLA into individual objectives that can be mapped into policies that can be executed. The SLO is responsible for that. The SLOs define metrics to enforce, police, and/or monitor the SLA. Some metrics that are being used are performance response time, component system availability (up time), and serviceability.

Traffic Conditioning is control functions that can be applied to a behavior aggregate application flow, or other operationally useful subset of traffic e.g. routing updates.

1.1.5 Policy Management

Policy management responsibilities are to manage and control the entry of packets into to the network, and define which services are available. To be able to implement the policy management we need a QoS policy server that would distribute, manage, and capture the network policy in the service provider's domains. A management system needs to be able to do the following:

- Create a policy
- Directory storage of policy information
- Policy server (distribution of the policies)
- Networks elements, which perform policy enforcement
- An application interface to all interaction between the policy elements and external applications

1.1.6 QoS Policies

To be able to enable QoS on the Internet we need policies to include preferential queuing or dropping, admitting or denying access, or encrypting the packet’s payload. Some protocols that support all these functions are:

- *COPS*

- *RADIUS RSVP*
- *IntServ*
- *DiffServ*

The ability of these protocols to successfully scale depends on the effectiveness of the network to administer and distribute consistent policy information to the multiple devices in the network, which perform the classification and packet conditioning or treatment. Protocols that are being used for distribution of the policy include LDAP, COPS, SNMP and TELNET/CLI. Some of these protocols will be discussed in greater detail on later sections of this thesis.

1.1.7 Qos Ranking

Table 1 shows the ranking list of the protocols based on the QoS support they offer.

QoS	Network	Application	Description
Most	X		Provisioned Resources end-to-end
	X	X	RSVP [IntServ Guarantee Services]
	X	X	RSVP [IntServ Controlled Services]
	X		Multi-Protocol Label Switching [MPLS]
	X	X	DiffServ.
	X	X	DiffServ or SBM
	X		Diffserv applied at network core ingress.
	X		Fair queuing applied by network elements (e.g. CFQ,WFQ,RED)
Least			Best effort service

Table 1: QoS Ranking

It’s obvious that RSVP can provide us with the most guaranteed QoS and Best-effort with the least guaranteed QoS support. As we will see later, RSVP does not scale well enough for use on the Internet. MPLS and DiffServ seem to be better solutions than RSVP and they seem to be making their way up. The worst of all these protocols in terms of QoS is the Best-effort, since it doesn’t offer any QoS control.

1.2 New protocols for IP QoS provision

As discussed earlier, there are a few protocols that aim to support IP QoS. Some of these have already failed to provide a scalable efficient service. Others are still investigated. A few of these protocols are the ReSerVation Protocol (RSVP), Integrated Services (IntServ), Differentiated Services (DiffServ) and the Multi Protocol Labeling Switching (MPLS). The two most promising protocols are MPLS and DiffServ [16]. The RSVP seems to be failing since it's very complex system and does not scale easily. RSVP provides a reservation setup through the routers. MPLS tries to solve the problem with the addressing of the IP protocol at the routers. The MPLS uses a 20-bit label to simplify the routing of the IP. MPLS is an independent protocol and can be complementary to DiffServ. It's expected that the use of MPLS with DiffServ may prove a good solution. MPLS resides in the routers.

1.2.1 Integrated services (IntServ)

The Integrated Services has been implemented to solve the problems we have today with the Internet. The Integrated Services aims to establish a QoS in the Internet and to enhance the Internet services, as was done in ATM. The main components of the Integrated Services architecture are the traffic control, traffic classes and the resource reservation setup protocol.

The Traffic Control consist of *Admission control, Packet classifier and Packet scheduler.*

The ***Admission control*** functions like a policeman. The Admission control checks the recourses of the network to decide whether it will make a new reservation or not. In this way it can also check to see if the connections use more resources from what are supposed to. Then accordingly the ISP can re-allocate bandwidth.

The ***Packet classifier*** is responsible to map the incoming packets into different classes. A class can be a single flow or a many flows.

The ***Packet scheduler*** is responsible for transmitting the packets streams according to the resources that have been reserved for them.

The IntServ architecture has 3 Traffic Classes. These three classes are the Guaranteed, controlled load, and Best-effort. By having these 3 classes we can categorize our users into these classes and charge them based on the class they use.

Guaranteed

The Guaranteed class guarantees the delay, bandwidth and packet loss. This class can be used for real-time application such as video, audio, etc.

Controlled Load

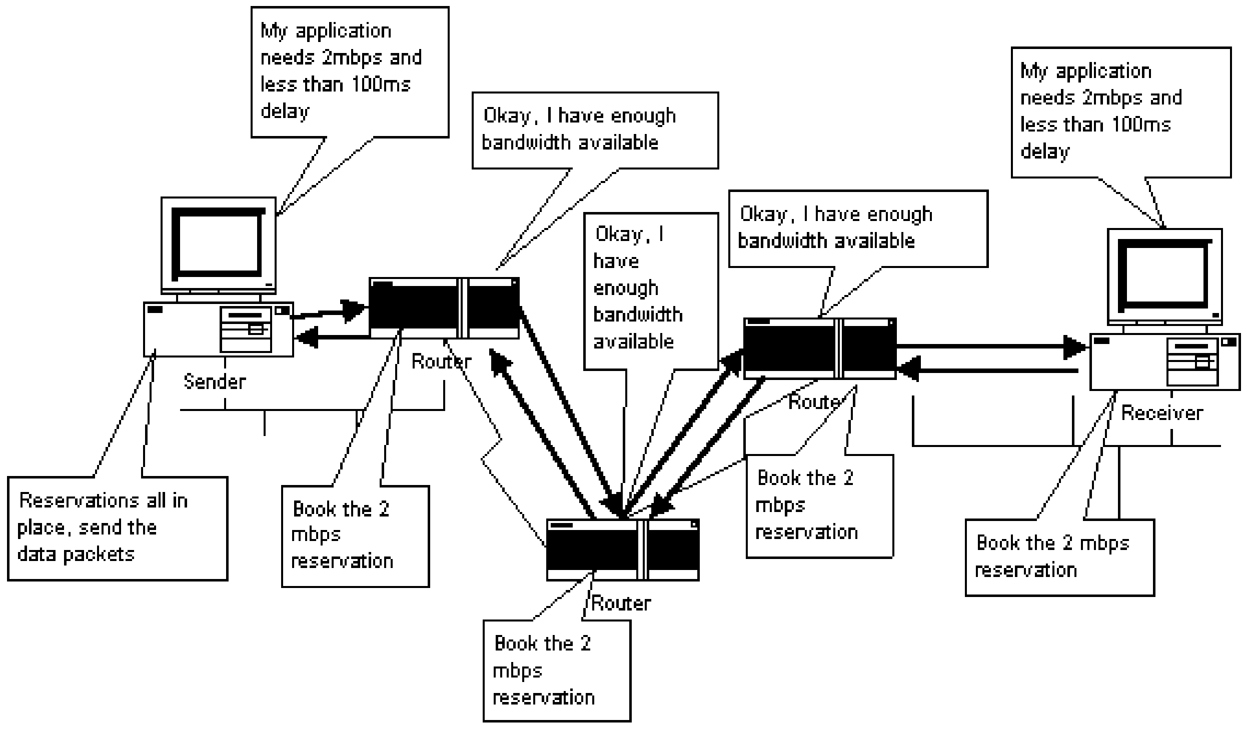
This class offers a better service than Best-effort but lower service than the Guaranteed class. It's mainly used for users who don't want to pay a lot of money for the guaranteed class, but also wants to get a better service than the average user. The packet losses and delays in this class will be minimized.

Best-effort

Best-effort will be all of the users who don't have strict quality of service requirements. This is the only class used in today's IP Internet. It's good for elastic applications, such as e-mail, and ftp.

1.2.2 Reservation Setup Protocol (RSVP)

The signaling protocol in the IntServ architecture is the RSVP. The RSVP is invoked when a request for a new reservation has been made. The source sends out the traffic requirements and will traverse along every node, which will check if it can obtain those resources, and sends it to the next hope, until it reaches the receiver. The receiver sends the reservation to the next node and passes it to the next node until reaches the source where the transmission starts. In case, one of the nodes can't allocate the resources that it has been requested from, it can announce the maximum resources that it can provide and the receiver will decide whether it can't accept it or decline it. In Figure 1 shows the steps of the RSVP procedures [17].



My application needs 2mbps and less than 100ms delay
Okay, I have enough bandwidth available
My application needs 2mbps and less than 100ms delay
Okay, I have enough bandwidth available
Okay, I have enough bandwidth available
Sender
Router
Router
Receiver
Reservations all in place, send the data packets
Book the 2 mbps reservation
Book the 2 mbps reservation
Book the 2 mbps reservation
Router
Book the 2 mbps reservation

Figure 1: RSVP Architecture

Disadvantages of RSVP

The RSVP is been already implemented in the Microsoft Windows 2000 server edition. The RSVP is been used for Intranets mostly but not for the

Internet. Some of the reasons that it has not been used in the Internet are the following:

- *Scalability*
- *Security*
- *Policy control*
- *Scalability*

The RSVP is a soft state protocol. This means that the RSVP has to refresh the state of each reservation. This requires higher CPU power and memory at the routers. The routers manipulate thousands of sessions that can be reserved by the RSVP; as an outcome is to cause delays on other critical applications.

Security

The RSVP doesn't provide any security to which nodes have authority to reserve network resources. In that respect the security on this protocol is not good enough to prohibit unwanted users to reserve more of what resources they are suppose to reserve.

Policy control

Again the RSVP doesn't have a good control to be able to policy that granted access to the resources.

1.2.3 Multi-Protocol Label Switching (MPLS)

Multi-Protocol Label switching (MPLS) is one of the three emerging technologies which support IP QoS. The MPLS approach will be the networking technology that delivers the traffic engineering capability and QoS performance for backbone networks to enable the support of differentiated services [9]. MPLS might solve the problems that IP networks face today, as for example deliver real-time applications, guarantee a certain QoS to the customer, and control the traffic over the network.

Forwarding and Routing

MPLS uses a label to route and forward the packet in the MPLS domain. This label is assigned by the ingress Label Switching Router. At the ingress of the MPLS domain the edge LSR functions like a classifier, and assigns a short fixed size label on each packet, based on the concept of forwarding equivalence classes, FEC. All packets belonging to one FEC take the same path and get the same treatment. After a packet has been assigned with a label is admitted in the MPLS domain where this label is been used to be routed accordingly. In the MPLS domain, the routers usually lookup the label of the packet and not the original packet to forward the packet to the appropriate router. At the egress point of the MPLS domain the edge router removes the label and forward the packet to the host. The major components of the MPLS network are shown in Figure 2.

The labels construct the Label Switched Path. The network administrators can direct traffic where they want by changing the LSP. There are two ways to establish the route for a given LSP: the control-driven, or the explicit route (ER-LSP).

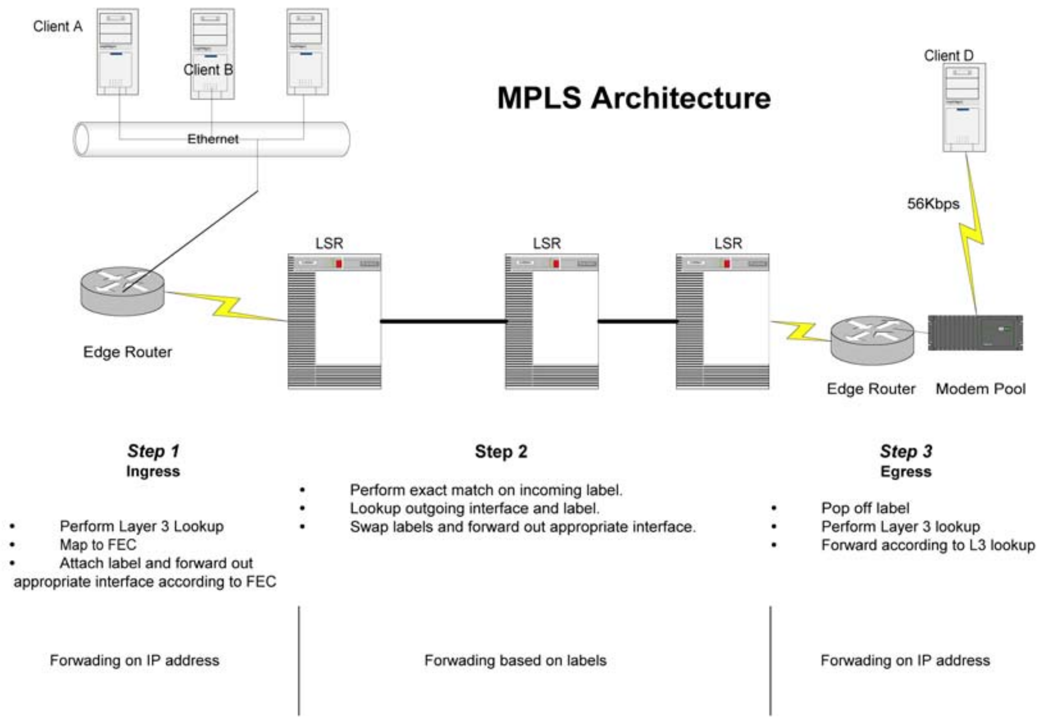

MPLS Architecture
Client A
Client B
Client D
Ethernet
56Kbps
LSR
LSR
LSR
Edge Router
Edge Router
Modem Pool
Step 1
Ingress
• Perform Layer 3 Lookup
• Map to FEC
• Attach label and forward out appropriate interface according to FEC
Step 2
• Perform exact match on incoming label.
• Lookup outgoing interface and label.
• Swap labels and forward out appropriate interface.
Step 3
Egress
• Pop off label
• Perform Layer 3 lookup
• Forward according to L3 lookup
Forwading on IP address
Forwading based on labels
Forwading on IP address

Figure 2: MPLS Architecture

In the case where we are setting up control driven LSP, each LSR determines the next interface to route the LSP based on its Layer 3 routing topology database, and sends the label request to the L3 next hop [9]. When setting up an ER-LSP, the route for the LSP is specified in the "setup" message itself, and this route information is carried along the nodes the setup message traverses [9]. In this case all the nodes along the ER-LSP will follow the route specification and send the label request to the next indicated interface .In this way the network administrators can manage and control the traffic engineering by using the ER-LSP. They can direct the traffic exactly where they want by specifying the exact nodes and interfaces the ER-LSP will traverse. Also, they can be less strict working on a higher level and not give all the details about the route. The labels of the packets have only local meaning in the MPLS domain. There are cases that we need to have more than one label for one packet. This is called label-stack. The label-stack uses the last in, first out stack that can contain as many labels as needed. This method is used for transmitting a label to a router that is not a direct neighbor.

Advantages of MPLS over Internet

A list of the Advantage of MPLS over the Internet is following:

- A router doesn't need to analyze the network layer packet header. The router can run a wide range of network layer protocols.
- Every packet that comes into the MPLS domain at the ingress router is assigned an FEC, forwarding equivalence class. This decision is made based on the packet header information or more information that the administrator wants to use.
- The edge routers require higher CPU and memory power because they do most of the work. The routers in the core they are cheaper and lower end routers since they just have to forward the packet based on the label.
- With MPLS the administrator has control over the engineering traffic. With the label packets can be forced to take a certain route through the network.
- The precedence or class of service (DiffServ) can be encoded in a label.

1.3 Introduction to DiffServ

Since 1997, a number of different approaches of implementing DiffServ networks have appeared in the literature [27] [29] [30]. These approaches are different in two ways: the high-level user perceivable services and the mechanisms required to achieve these services. In 1998, a working group for Differentiated Services (DiffServ WG) had been established. The main goal of this group is to standardize the use of Type of Service in both IPv4 and IPv6.

DiffServ exploits the ToS (Type of Service) field in the IPv4 packet header to provide rudimentary QoS to the users, see Figure 3. Briefly, DiffServ provides a classification or differentiation of classes among the users. By classifying the users in different classes you can provide them with better (prioritized) QoS. All packets belonging to the same class are treated the same way. DiffServ uses the 6 bits of the 8-bit ToS field that it has been renamed to DS (Differentiated Services field). The other two bits are reserved for future use; see Figure 3.

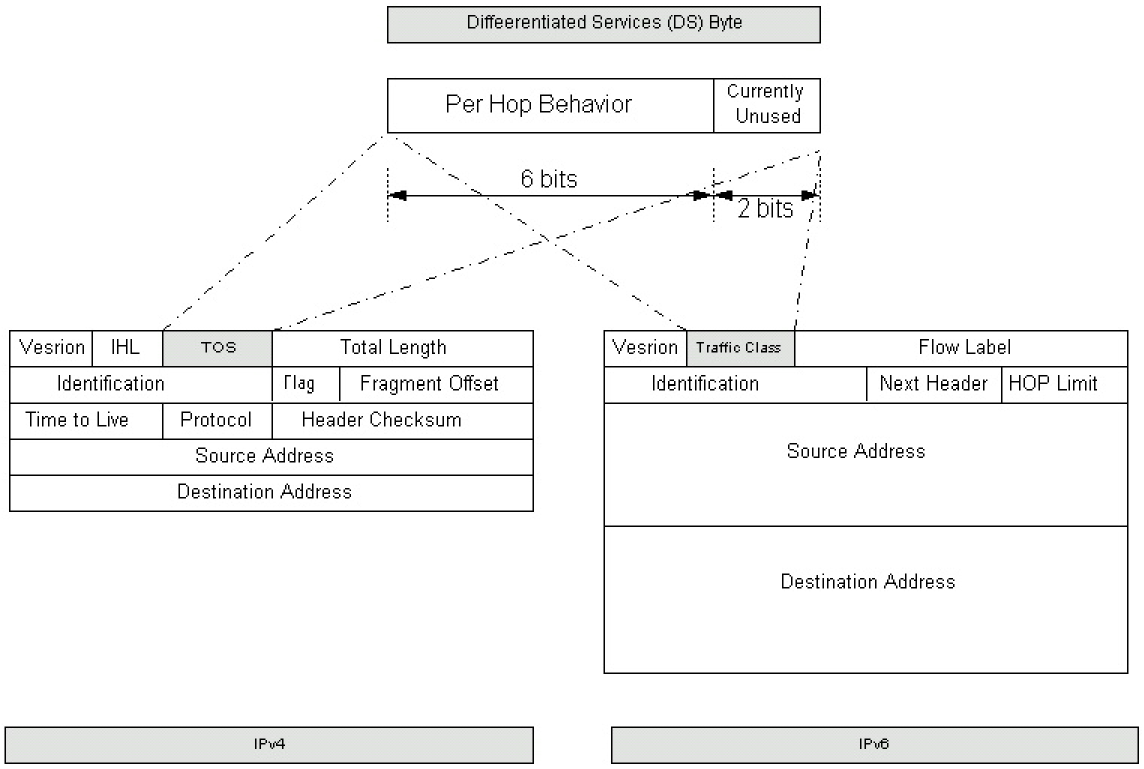


Figure 3: DS Byte in IPv4 and IPv6

CU=currently unused (2-bits)

DSCP= Diff-Serv code point (6-bits)

DSCP=101100: EF (Expedited Forwarding)

DSCP=000000: DE (Best effort)

others still under study

DiffServ appears to be a promising architecture for providing differentiation of service to aggregated users. It has received a lot of attention in the literature and lately some implementations are appearing, as for example in Linux implementation [6], and commercially on some routers [14]. In this thesis we will focus on the DiffServ architecture, details of which appear in the later chapters.

2 Differentiated Services (DiffServ)

2.1.1 Differentiated Services Model

Figure 4 shows the Differentiated Services Model. The DiffServ domain is broken down to boundary nodes and interior nodes. The boundary nodes are responsible for setting the DS bits in the packet, and the conditioning of packets. The interior nodes are responsible for forwarding packets in different ways based on the DS field. In order to have consistent service you must have common rules. The rules are used to set the bits of the DS field code points and how the packets are conditioned at the boundary nodes. Rules also define how the packets are forwarded inside the network at the interior nodes.

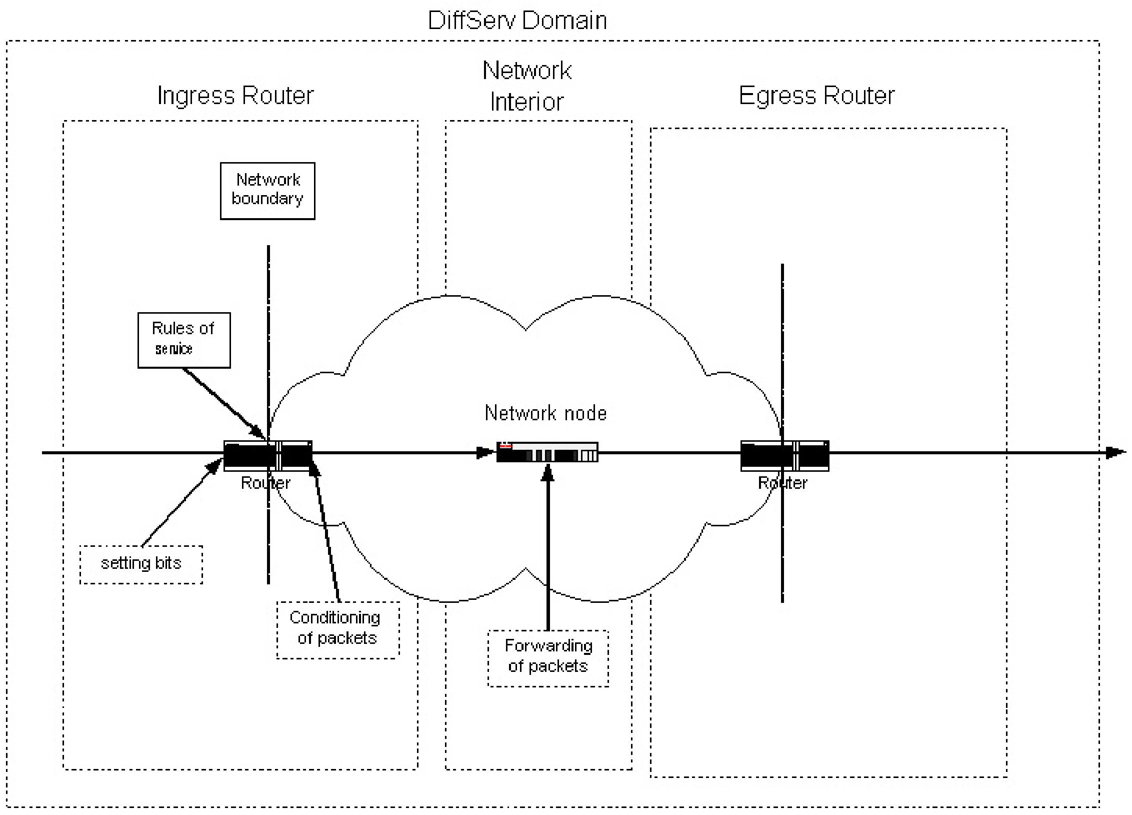


Figure 4: DiffServ Domain

2.1.2 Terminology

Some terminology is necessary to be explained for better understanding of Differentiated Services.

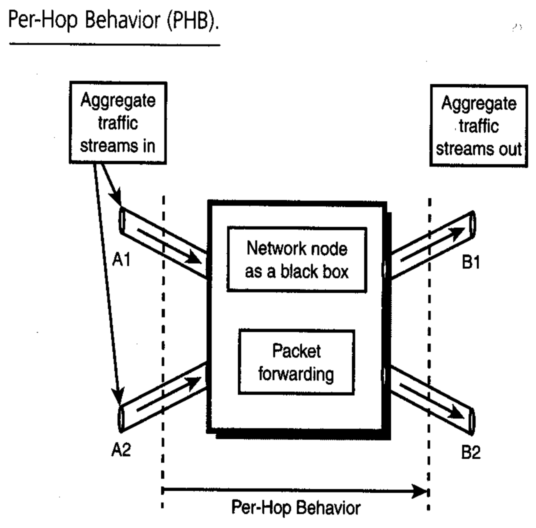
Per-Hop Behavior (PHB)

PHB denotes a combination of forwarding, classification, scheduling and drop behaviors at each hop. The main purpose of PHB is to make a comprehensible connection between packet-level implementations and service models [3].

Some of guidelines for designing a PHB are the following:

- PHB is primarily a description of desired behavior on a relatively high abstraction level; in particular, a PHB must have a comprehensible motivation.
- PHB should allow the construction of predictable services.
- The desired behavior should be externally observable.
- The desired behavior should be local-that is, it should concern behavior within one node rather than the whole network.
- The description of behavior is related to an aggregate that consists of all packets belonging to the same PHB in a certain point of the network.
- The PHB description should not suppose any particular conditioning function at the network boundary.

The traffic conditioning and service provision functions must be separated from forwarding behaviors (RFC 2475). The reason of the separation of the traffic conditioning and forwarding is flexibility, see Figure 5.


Per-Hop Behavior (PHB).
Aggregate traffic streams in
Aggregate traffic streams out
Network node as a black box
Packet forwarding
A1
A2
B1
B2
Per-Hop Behavior
Figure 5: Per-Hop Behavior

PHB class

A PHB class is a collection of PHBs intended to be applicable for transmitting packets of one application. The packets shouldn't be reordered inside the network. The PHB class with the appropriate traffic conditioning functions is the nearest equivalent for the network services in connection-oriented networks.

Codepoints

Codepoints are the 8 bits that used to inform the interior nodes about the PHB of the packet. Several different codepoints can map to the same PHB.

Mechanisms

Mechanism is the implementation of one or more Per-Hop Behaviors according to a particular algorithm. A mechanism can be used for implementing several PHBs, and several mechanisms are usually needed to implement a PHB. Figure 6, shows the main building blocks of DiffServ.

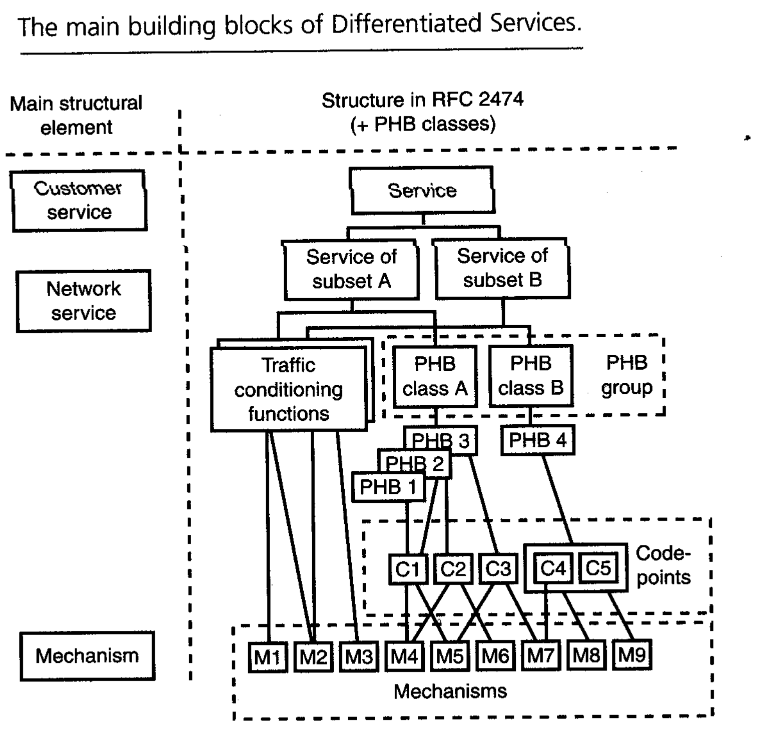


Figure 6: Main Blocks of DiffServ Services

2.2 DiffServ Architecture

The RFC 2475 defines the Architecture for Differentiated Services. Mostly the RFC2475 talks about the scalability based on the DS field. The service characteristics may be specified in terms of throughput, delay, jitter, loss, or relative priority of access to network resources. The PHBs are developed based on the above characteristics.

The main requirements of a basic architecture of a DiffServ Services are the following:

Versatility: A wide variety of end-to-end services should be possible to realize; network services should be independent of applications, and they should be directly applicable with current applications and with current network services.

Simplicity: The overall system or parts of it should not depend on signaling for individual applications. A small set of forwarding behaviors should be necessary.

Cost efficiency: Information about individual flows or customers should not be used in core nodes, but only states of aggregated streams should be used in core nodes.

2.2.1 Architecture Model

This section focuses on the architecture model of the Differentiated Services. For better understanding of the architecture model, we need to clarify some more terminology. Figure 7 shows the basic elements of Differentiated Services Network. A list of the basic elements of DiffServ is the following:

- **Boundary node:** A collection of functions needed to interconnect a DS domain to another DS domain or to non-DS-capable domain.

- **Interior node:** A collection of functions needed if a node is connected only to other DS-capable nodes.

- **Ingress node:** A collection of functions needed to handle incoming traffic streams to a DS domain.

- **Egress node:** A collection of functions needed to handle outgoing traffic streams from a DS domain.

In reality, the boundary node can be a boundary node for some traffic stream and an interior node for some other streams. An interior node may have a limited capacity of traffic conditioning.

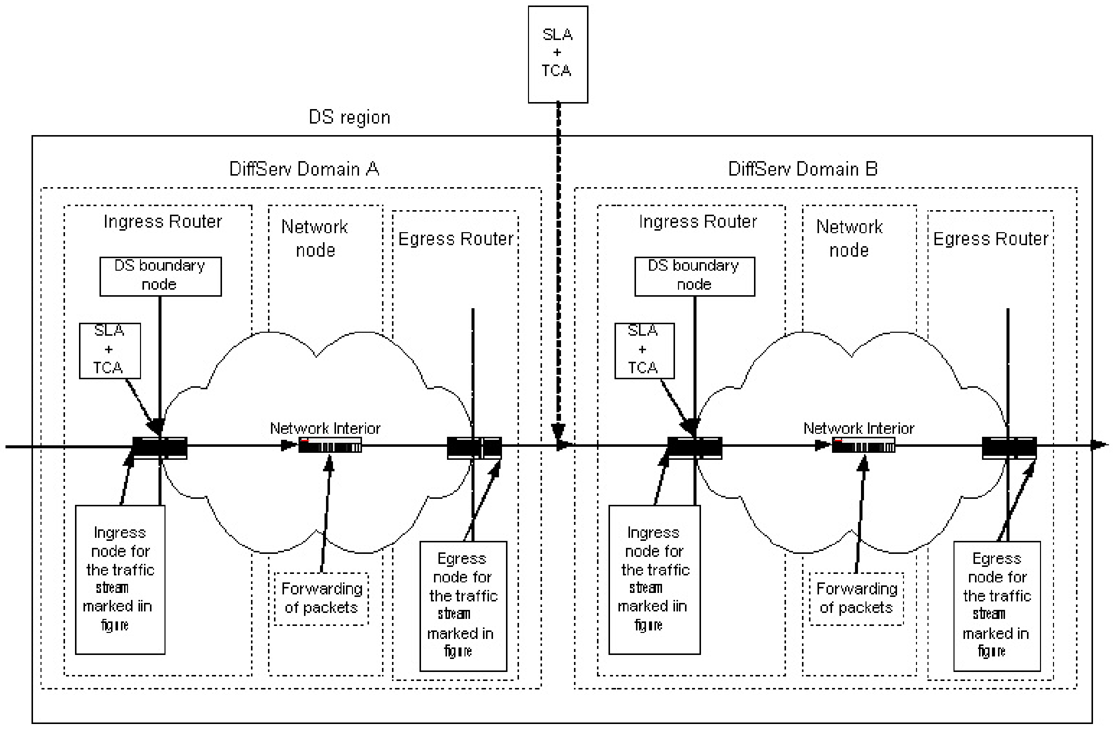


Figure 7: Basic elements of a Differentiated Services network

At the boundary nodes takes place the traffic condition based on the Service level Agreements. There are two level agreements.

- **Service-level agreement (SLA):** A contract between a customer and a service provider that specifies the forwarding service

- **Traffic-conditioning agreement (TCA):** Defines the rules used to realize the service, such as metering, marking, and discarding

2.2.2 Traffic Classification and Conditioning

Figure 8 shows the logical structure of traffic classification and conditioning functions. Traffic conditioners are usually located at DS boundary. The classification is made according to the source-destination and DS filed. A traffic profile is one way to present the traffic-conditioning rules. The packets can be either in-profile or out-of-profile, based on the results at the arrival time of the packet. The in-profile packets have higher priority over the out-of profile packets. The traffic meter measures each traffic stream.

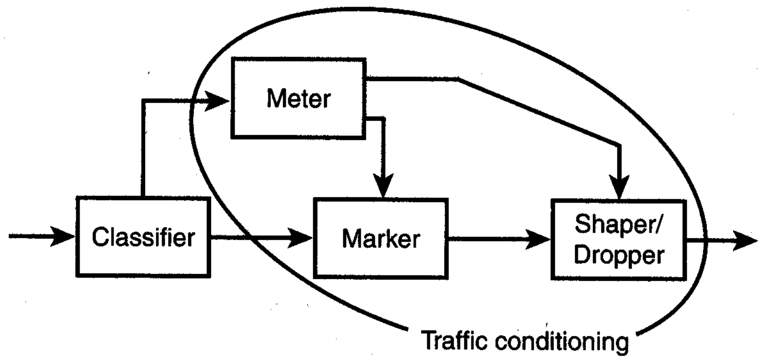


Figure 8: Packet classifier and traffic conditioning according to the RFC2475

Traffic meter informs the marker, shaper and dropper mechanisms about the state of the stream:

- **Marker:** Sets an appropriate codepoint to the DS field of the packet.

- **Shapers:** Used to smooth the traffic process of particular aggregate streams
- **Dropper mechanisms:** Based on the SLA and TCA, some packets can be discarded at the traffic-conditioning element.

2.3 Per-Hop Behavior Groups

This section describes the per-hop behavior groups. It concentrates on the following four PHB groups:

- Class Selector PHB
- Assured Forwarding
- Expedited Forwarding PHB
- Dynamic RT/NRT PHB

2.3.1 Class Selector PHB

The Class Selector PHBs is been defined for backward compatibility for Ipv4 TOS octet. There is some usage of the 0-2 bits of the TOS of Ipv4 that were intended for the Department of Defense applications. The RFC 2474 states the following:

> A class Selector PHB should give packets a probability of timely forwarding that is not lower than that given to packets marked with a lower Class Selector PHB, under reasonable operating conditions and traffic loads.

The CS PHB is situated for Resource Sharing Model. Figure 9 shows an implementation of Class Selector PHB. The first two queues are high priority queues and they accept queues as long as they have space. The lowest queue is divided in thresholds. The lowest queues could be RED.

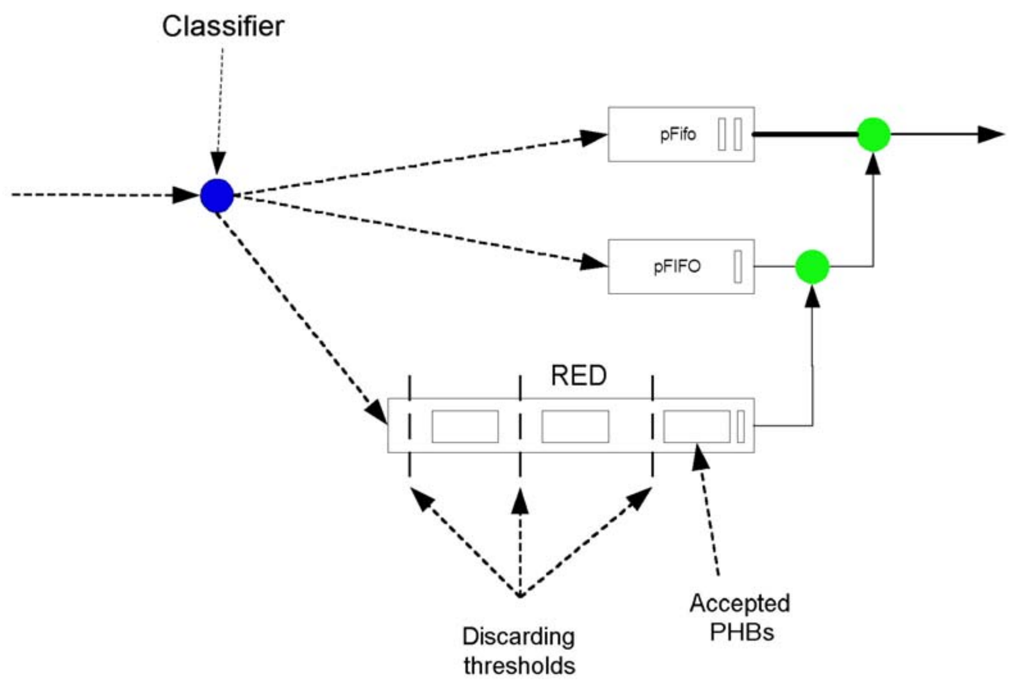


Figure 9: Class Selector PHB Implementation

2.3.2 Assured Forwarding (AF)

The assured forwarding (AF) has four classes and within each class 3 drop-precedence. Any packet exceeding their profile will be demoted but not necessarily dropped. Every node that supports AF must at least implement these four classes. In AF every node must reserve a certain amount of resources such as bandwidth, buffer size and etc. Every packet that enters at the edge router is subject to traffic conditioning. At the edge router the packets can be dropped, shaped, reassigned to another class or to higher or lower drop precedence. After the packet is in the network it just forwarded to the next router. With AF PHBs have the flexibility to implement different service models based on applications, individual's customers, or organizations. Figure 10 shows an implementation of AF PHB.

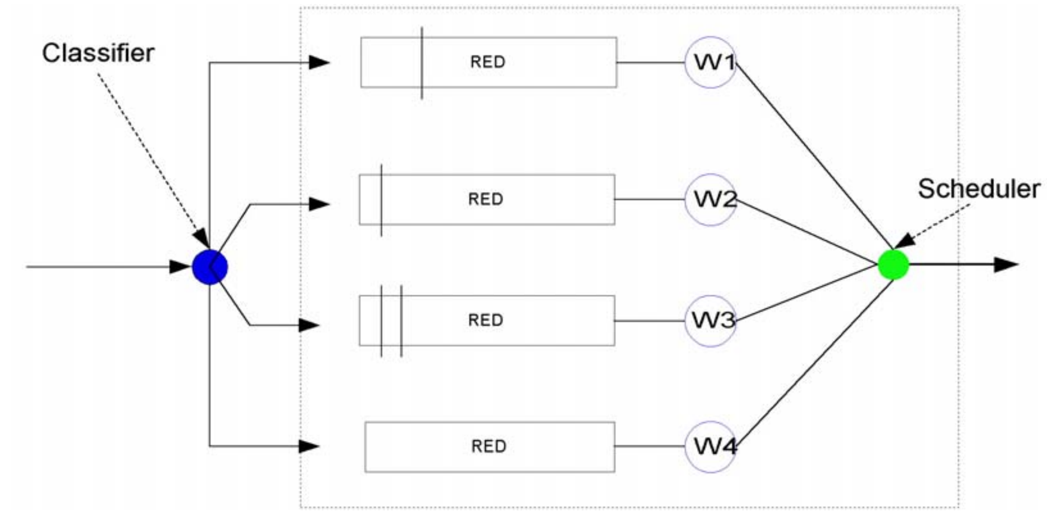


Figure 10: AF implementation based on four queues

2.3.3 Expedited Forwarding (EF)

The Expedited Forwarding minimizes the delay, loss and jitter. In the EF if the packet exceeds its profile will be discarded. In order to keep the loss, delay and jitter low the packet should see no queues. The EF uses a single bit to indicate that it is high priority [3]. The EF guarantees the minimum departure rate at every node. The network administrator can set the minimum and maximum departure rate from every node. If the packets exceed the maximum departure rate then it discarded so it doesn't damage any other traffic.

The classification takes place at the ingress router. For every packet that comes in the ingress router, the router classifies the packet according to its SLA (Service Level Agreement). After the packet has been classified then the rest of the routers can use the DS field to forward the packet to its destination, with the appropriate priority. There is no marking at the EF PHB since there is only one level of importance. In case the packets arrive before its scheduled time there are three options at the boundary and interior nodes:

- To forward the packet immediately
- To forward the packet at the scheduled time
- To discard the packet

The EF PHB can implement a leased line service as a primary model and guaranteed connection as a secondary service model. An implementation of EF PHBs is shown at Figure 11.

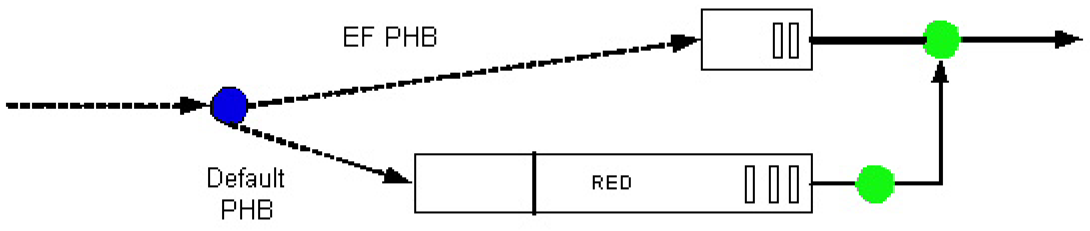


Figure 11: Expedited forwarding Implementation

Figure 11 show a small queue with strict priority and a default queue with RED mechanism. This is because we want to minimize the RTT and the delay. Keep in mind that in case we are transmitting a real time data they are useless if the data exceed a certain delay.

2.3.4 Dynamic RT/NRT PHB Group

The DRT-PHB contains two classes and six PHBs. Figure 12 shows the classes. The PHB classes offer two distinctly delays. One delay is for the real time applications such as videoconferencing, IP telephony and etc. The second delay is for elastic applications such as email, ftp and etc. Six importance levels offer wide dynamics for various traffic-control. The two delays and the six-importance level can be increased.

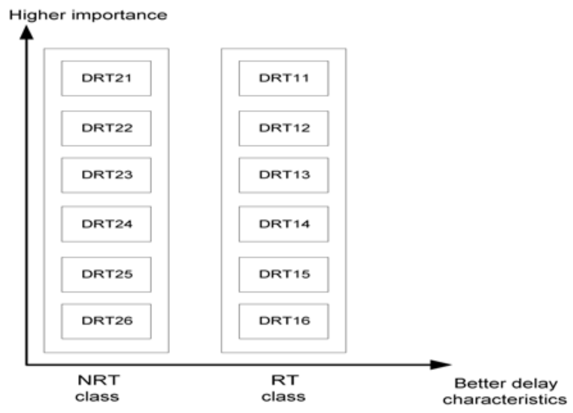


Figure 12: Structure of DRT-PHB group

The DRT-PHB group has the flexibility to be applied to any of the three service models: application, customer, or organization model. This flexibility is gained because the DRT-PHB group uses the nominal bit rate, NBR. NBR defines the relative amount of resources that a certain entity is supposed to achieve from the network. An implementation of the RT-PHB group is shown at Figure 13.

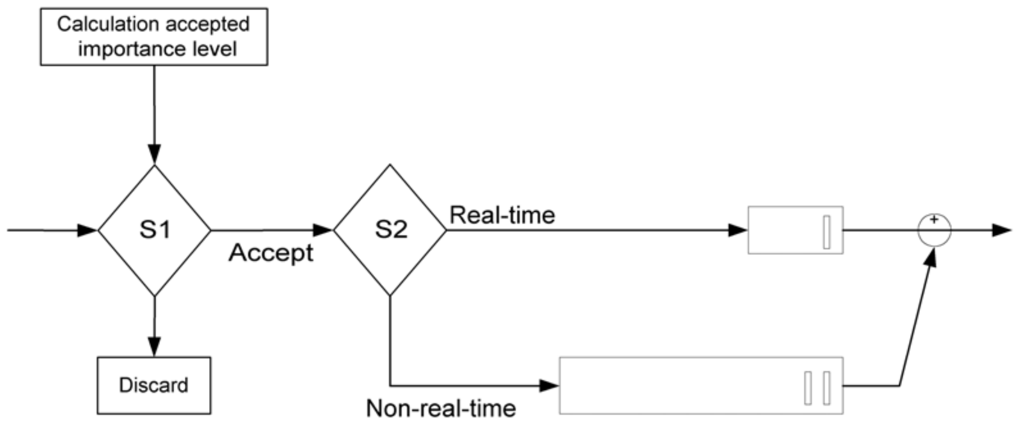


Figure 13: Implementation of the DRT-PHB

3 Traffic Management in DiffServ

In order to deliver differentiated services, it is necessary to offer the means to manage traffic. In a DiffServ setting one can identify a number of alternatives, such as Class Selector PHB, Assured forwarding Group, Expedited Forwarding PHB and DRT/NRT PHB. Next we discuss the terms urgency and importance and then detail some of the mechanisms than can be employed, with special emphasis on the LINUX implementation.

3.1 Urgency and Importance

Urgency and importance are very important terms for traffic handling. What do we mean when we say this packet has a high urgency? A packet with high urgency must be delivered as soon as possible with as small delay as possible. Of course there are many combinations of urgency and importance. A packet can be urgent and important, urgent but not important, important but not urgent, or not urgent and not important.

Real time applications such as IP telephony and videoconferencing require a small urgency otherwise their data can be useless.
Importance on the other hand can be used to differentiate certain packets over others. For instance if we wanted to give a higher priority to a telnet application over email we could do that by using importance characteristics. We could mark all the telnet packets with higher importance and at the event of a congested network the email packets will be discarded first before the telnet packets [3]. Figure 14 shows the scenario based on important versus less important. We can see from the figure that in case of one individual flow there is a higher probability to drop an important packet rather the non-importance. At aggregated flows, there are more chances to drop a non-important packet since there is a higher chance in that time slot to have non-important packets.

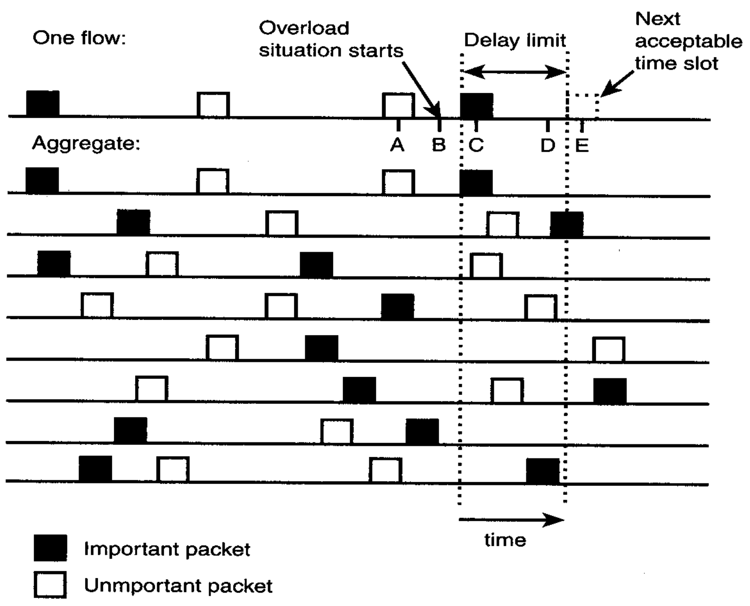


Figure 14: Selection of Packets

3.2 Traffic Management in Boundary nodes

The traffic handling can be broken down into four phases:

1.Setting the target
2.Collecting information
3.Making the decision
4.Executing the decision

3.2.1 Classifiers

A classifier is a mechanism used to select the PHB class for a traffic flow. There are various models that can be used to classify a PHB class such models are the following:

- The user selects a definite service class from the available classes.
- The application automatically selects a preferable service class for each flow or packets.

- The network selects an appropriate service class based on information about the application.
- The network selects an appropriate service class based in the customer contract regardless of the application.
- A combination of the first four approached.

The first approach is not very practical to implement, since it requires additional mechanisms to allow the simultaneous use of several classes, such as IP telephony and data. The second approach seems more practical, to have the application to select a service class. In order to be able to implement this scheme the customer and the service provider have to use the same DS codepoints. The problem is that the classification must be made at the customer premises and might not have the equipment for it.

The third approach seems the more reasonable in the case the customer doesn't have the equipment. The fourth approach is applicable and reasonable by using SLAs between the provider and the customers. The packet classifiers are broken down into two types, the behavior aggregate, BA, classifier and multi-field classifier, MF.

3.2.1.1 Behavioral Aggregate Classifier

BA classifies or selects packets based on the DS field only. It's used mostly on the interior routers because it's very difficult to classify packets for customers, since it classifies packets based on the DS field.

3.2.1.2 Multi Field Classifier

As we have seen at the BA classifier is mostly used for interior routers, a multi field classifier is used at the boundary of a DS domain. The MF classifier selects or classifies packets on the header of the packet.

3.2.2 Meters

The traffic-metering module is responsible for sorting the classified packets into the right importance level. One way to do this, the packet marking must take into account several measuring results. Another way is that the marking, shaping, and dropping decisions must be taken based on the measuring result of the class to which the packet belongs.

3.2.3 Packet Marking

The main objective of the packet marking is to map packets into one of the available importance levels of the PHB class used by the flow. There are two marking principles:

- When a packet exceeds a threshold, it is marked as low importance, but it is not used to determine the load level of the following packets. Effectively, the allowed bit rate of the higher importance level is totally independent of the load of lower level importance level.

- When the momentary load level exceeds a threshold, every packet is marked with lower importance.

3.2.4 Traffic Shaping

The shaping module is responsible for remarking the packets to lower importance level. The user has the freedom to shape its traffic before it sent to the network.

3.2.5 Packet Dropping at Boundary Nodes

In case the customer uses leased-line or guaranteed connections services, it may require that nonconforming packets be discarded immediately.

3.3 Traffic-Management Functions in Interior Nodes.

There are some differences between the interior router and the boundary router. The main parts of the interior routers are the buffering and discarding. At the interior nodes the classification is based on the DSCP field of packet.
There are many different queuing systems that are available for buffering such as FIFO,SFQ, CBQ, RED, and etc.

3.3.1 Queuing Disciplines

3.3.1.1 Pfifo_fast

PFIFO stands for packet First In, First Out. Also know as First Come First Served (FCFS) queuing. There is only one queue and all the packets are treated equally. The default size of the PFIFO queue is 100 packets in Linux operating system. The queue contains three bands, 0, 1 and 2. All the bands must comply with the FIFO rules. The bands are processed based on their priority. Band 0 has the highest priority and band 1 has priority over 2. In order to process band 1, band 0 must be empty. PFIFO stores the packets when is congested and forward them based on the arrival time [14].

PFIFO will not give priority to high priority packets over low priority packets. Ill-behaved sources can exploit most of the bandwidth with the result that important traffic will be dropped at the expense of lower priority traffic. At the event of congestion, when the queue fills up the PFIFO will drop all the packets. PFIFO is very suitable for large links that don't have large delays and minimal congestion.

3.3.1.2 Priority Queuing

Priority Queue, PQ, allows to configure four traffic priorities. This can be done by using several filters in series. The packets will be placed to the appropriate queues based on the header characteristics of the packets. The queues with highest priority is dequeued until it's empty and then move to the next queue. Every time a packet is transmitted, the queues are scanned based on their priorities and start it’s transmition.

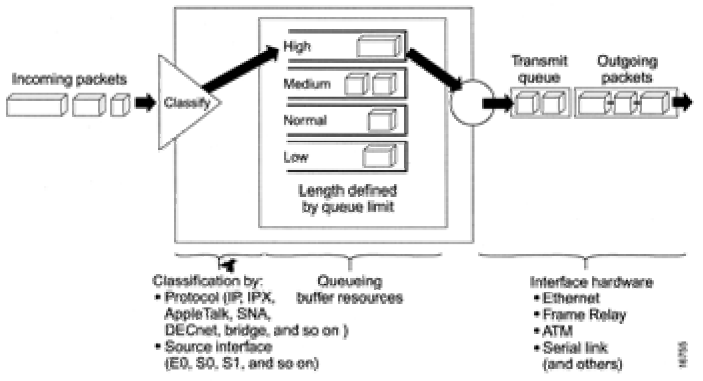

Incoming packets
Classify
High
Medium
Normal
Low
Length defined by queue limit
Transmit queue
Outgoing packets
Classification by:
• Protocol (IP, IPX, AppleTalk, SNA, DECnet, bridge, and so on)
• Source interface (E0, S0, S1, and so on)
Queueing buffer resources
Interface hardware
• Ethernet
• Frame Relay
• ATM
• Serial link (and others)

Figure 15: Priority Queuing

Packets can be classified based on the following list:

- Protocol or sub protocol type
- Incoming interface
- Packet size
- Fragments
- Access List

3.3.1.3 Custom Queuing

Figure 16 shows how the custom queuing works. CQ dequeues the packets in a round robin fashion. CQ allows specifying the number of packets or bytes each queue will be transmitting. In this way, CQ allocates the bandwidth among the queues. For every network interface the CQ maintains 17 queues. The queue number 0 has the highest priority of the other 16 queues. The system queue number 0, services the keep alive packets and signaling packets. CQ is statically configured and cannot be configured dynamically.

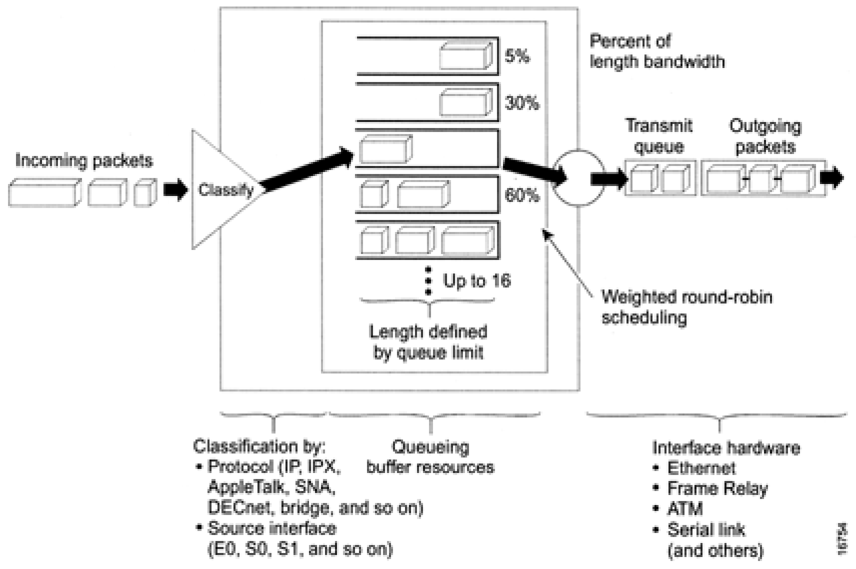

Percent of length bandwidth
5%
30%
60%
Transmit queue
Outgoing packets
Incoming packets
Classify
Up to 16
Length defined by queue limit
Weighted round-robin scheduling
Classification by:
• Protocol (IP, IPX, AppleTalk, SNA, DECnet, bridge, and so on)
• Source interface (E0, S0, S1, and so on)
Queueing buffer resources
Interface hardware
• Ethernet
• Frame Relay
• ATM
• Serial link (and others)
16754

Figure 16: Custom Queuing

3.3.1.4 Stochastic Fairness Queuing (SFQ)

SFQ was proposed by McKenney. SFQ is a simple implementation of fair queuing algorithms family. The incoming packets are classified based on the source-destination address and port number. This is achieved by using a simple hash function to map the incoming packet to an available queue. The classification of the incoming packet to the queue is probabilistic. Different flows can reside in the same queue despite their importance. The hash function changes periodically in order to avoid packets coming from the same source to reside in the same queue.

Flow is the sequence of data packets having enough common parameters to separate them from other flows. SFQ consists of dynamically allocated number of FIFO queues [4]. Based on McKenny, an SFQ may need to have 5 to 10 times more queues than the active source-destination pairs. The SFQ runs in a round robin, sending one packet from each FIFO in one turn. In table x, you can see the probability that one flow can share a queue with other flows. SFQ can divide the bandwidth exactly among all active queues and that the bandwidth of a queue is divided exactly evenly among flows directed to it. The benefits of SFQ are that requires little CPU and memory usage.

3.3.1.5 Weight Fair Queuing

WFQ provides dynamic fair queuing to the entire network by dividing bandwidth across queues of traffic based on weights. WFQ is a flow-based algorithm that simultaneously schedules interactive traffic to the front of a queue to reduce response time [14]. Most variants of the WFQ discipline are compared to the Generalized Processor Sharing (GPS) scheduler, which is a theoretical construct, based on a form of a processor sharing.

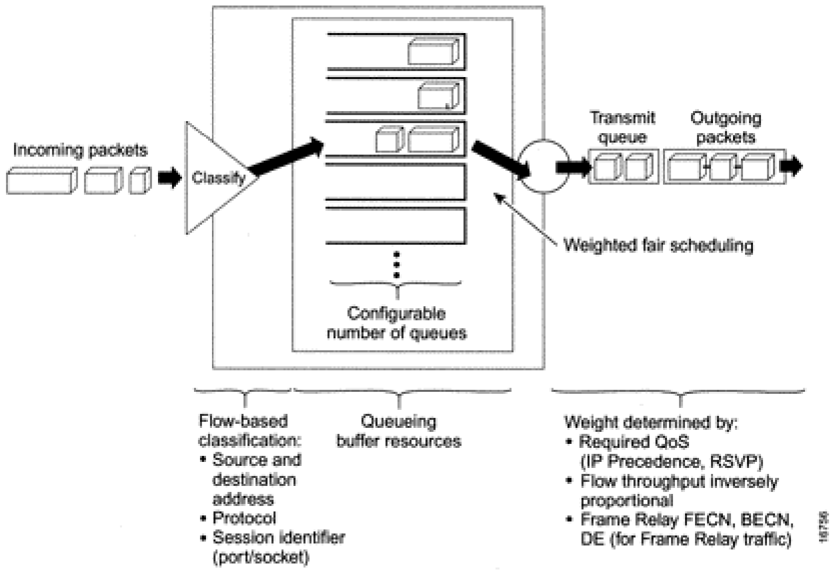

Incoming packets
Classify
Transmit queue
Outgoing packets
Weighted fair scheduling
Configurable number of queues
Flow-based classification:
• Source and destination address
• Protocol
• Session identifier (port/socket)
Queueing buffer resources
Weight determined by:
• Required QoS (IP Precedence, RSVP)
• Flow throughput inversely proportional
• Frame Relay FECN, BECN, DE (for Frame Relay traffic)
16756

Figure 17: Weighted Fair Queuing

Figure 17 shows the WFQ architecture. WFQ provides traffic priority management that dynamically sorts traffic into messages that make a conversation. WFQ breaks up the train of packets within a conversation to ensure that bandwidth is shared fairly between individual conversations and that low-volume traffic is transferred in a timely fashion. The classification of traffic is based on packet header addressing such as source and destination network or MAC address, protocol, TOS and etc. In WFQ there are two categories of flows: high-bandwidth sessions and low-bandwidth sessions. Low bandwidth has a higher priority over the high-bandwidth session.

The order of removal from the fair queues is determined by the virtual time of the delivery of the last bit of each arriving packet. WFQ is aware of the IP presence of the packet. In other words WFQ detects higher priority packets marked with precedence by the IP forwarder and can schedule them faster. As the precedence increases, WFQ allocates more bandwidth to the conversation during periods of congestion. WFQ uses weights to determine the order of the queues that are emptied. First serves the queues with the lower weights.

3.3.1.6 Random Early Detection

The Random Early Detection was proposed by Sally Floyd and Van Jacobson. The basic idea of the RED is to calculate the average queue size and if the average exceeds a certain threshold the incoming packets are dropped randomly based on the probability that depends on the average queue size. RED increases the fairness over the previous method, the drop tail method.
The RED can be used with Explicit Congestion Notification (ECN). In the case that we use ECN with RED instead of dropping the packets we mark them. If the queue gets full then it will drop the packets, see Figure 18.

The ECN notifies the TCP sources by suing some bits in the TCP header. Then the TCP sources reduce their sending rate, by doing this we are avoiding a congestion state. Red can keep the queue size low if we use the correct parameters.

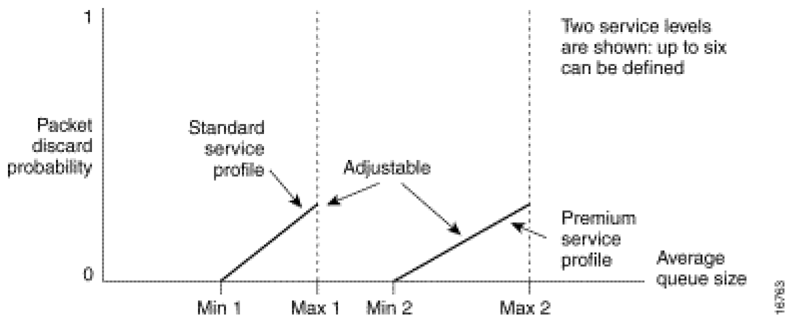


Figure 18: RED Packet Drop Probability

Disadvantages of RED

In the case where we have a few TCP sources using one link and the TCP source reduces its sending rate when it experiences a packet loss it will lead to underutilization of the link. In case there are many TCP sources only few sources may be reduced and the behavior of red will be similar to drop tail. These problems have been solved with the Adaptive RED and BLUE.

3.3.1.7 n-RED

The n-RED is the expansion of the RED. In n-RED we have two classes of traffic, called IN and OUT. This queue has been named as RIO, RED IN and OUT. The RIO has the following two sets of parameters:

1st set:
    Thresholds
    Drop probability

2nd set: Variables
    Average queue size

There are two averages of sizes that needed to be calculated in RIO. The first average of queue size is for the drop probability of the IN-packets. This average queue size counts only the IN-packets that are coming in the queue. The second average queue size for the OUT-packets is calculated based on the total queue size (OUT- and IN-packets). In other words the second average queue for the drop probability it counts and the IN and OUT-packets. The RIO queue uses two RED queues therefore is called 2-RED. We can implement n-RED queues as far we have n sets of parameters and variables.

3.3.1.8 Weighted Random Early Detection

The WRED uses the RED algorithm and the IP Precedence to provide for preferential traffic handling of higher priority packets. The WRED at a congested point can drop lower priority packets and give priority to the preferable classes. The IP Precedence controls which packets are dropped [14]. For instance traffic that has lower precedence has a higher drop rate. In Figure 19, we can see a diagram of the WRED and how it works. WRED avoids the problem of the globalization and tries to make an early detection of congestion as it also provides for multiple classes of traffic. The WRED is used on the core routers rather on the network's edge. The WRED gives the flexibility to the network administrator to assign a weight to the IP precedence, as he/her believes is better for its network.

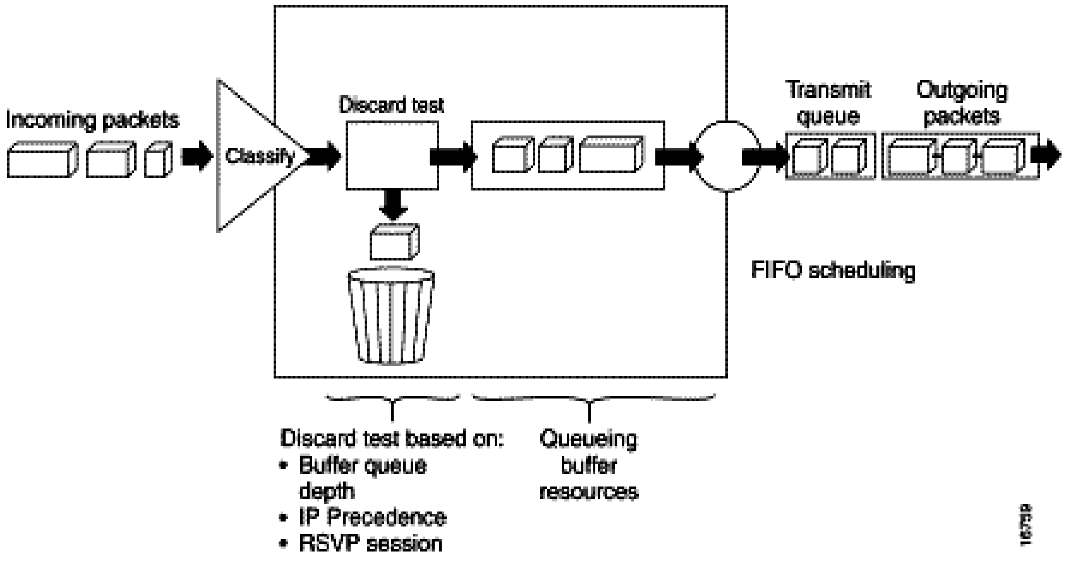


Figure 19: Weighted Random Early Detection

The WRED drops packets selectively based on the IP Precedence. It works on the notion that if the packet has a high IP Precedence then it's very highly to be delivered to its destination. Packets with lower IP Precedence will be dropped first. The WRED starts dropping packets as soon it see the queue to start getting congested in order to prevent the congestion. By doing this avoids the global synchronization because it will not need to drop very large packets at once. Users who are sending a lot of traffic over the network are more likely that their sending rate will be reduced in comparison with the users who are not sending so much traffic.

3.3.1.9 Class Based Queuing

Class Base Queering is another queuing discipline that solve the resource denial problem that we could have with other disciplines. In other words, CBQ can prevent classes from starvation. The CBQ is based on the link-sharing concept [1]. In a non-congestion state at the leaf classes, CBQ uses a general scheduler.

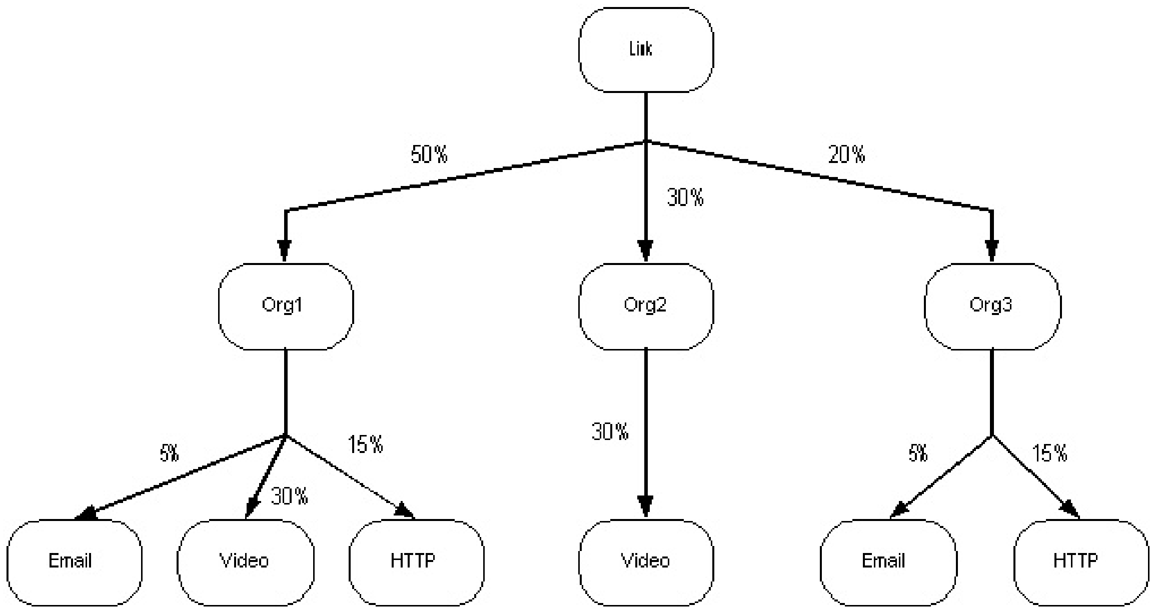


Figure 20: CBQ link share structure under no congestion

At the event that the classes become congested a link share scheduler is activated see Figure 20. This scheduler is responsible for isolating the traffic among the classes. CBQ has several parameters that can isolate, borrow or bounded traffic among the classes. This can be done from the top-level stage, see Figure 21. The general scheduler within a priority class is freely chosen. Implementations of CBQ use weighted round robin (WRR) and packet-by-packet round robin (PRR).

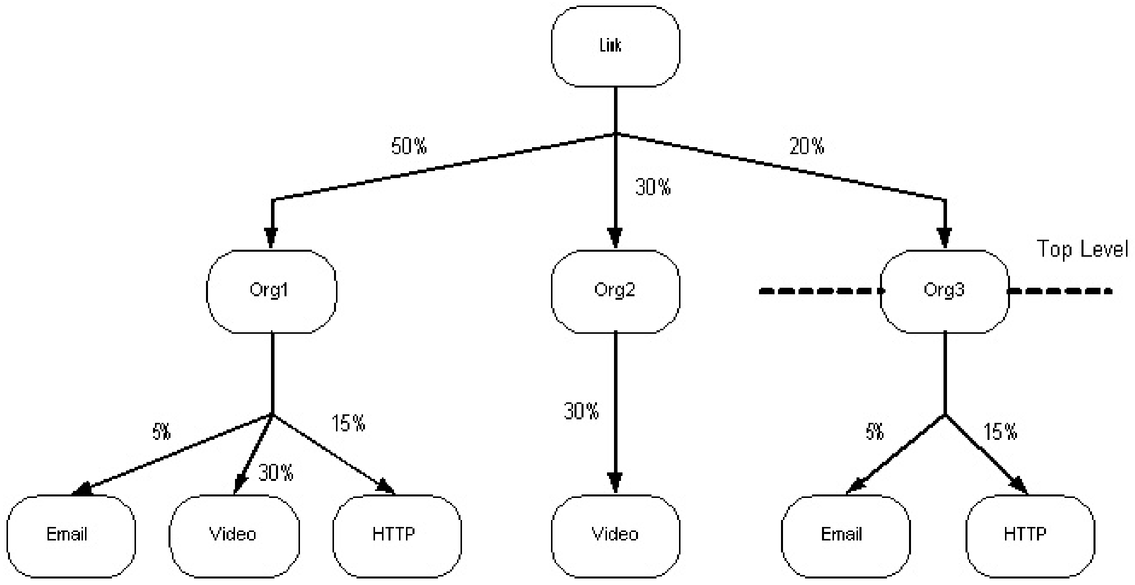


Figure 21: CBQ link share structure under congestion

3.3.1.10 CSZ Scheme

The CSZ objective is to isolate the link capacity to different traffic classes. In CSZ guaranteed service is provided by WFQ scheduler. WFQ assigns a share of link capacity to each flow. WFQ assigns a share of link capacity to each active flow. The predictive service in CSZ is a provided by priority queue. Figure 22 shows the CSZ scheme.

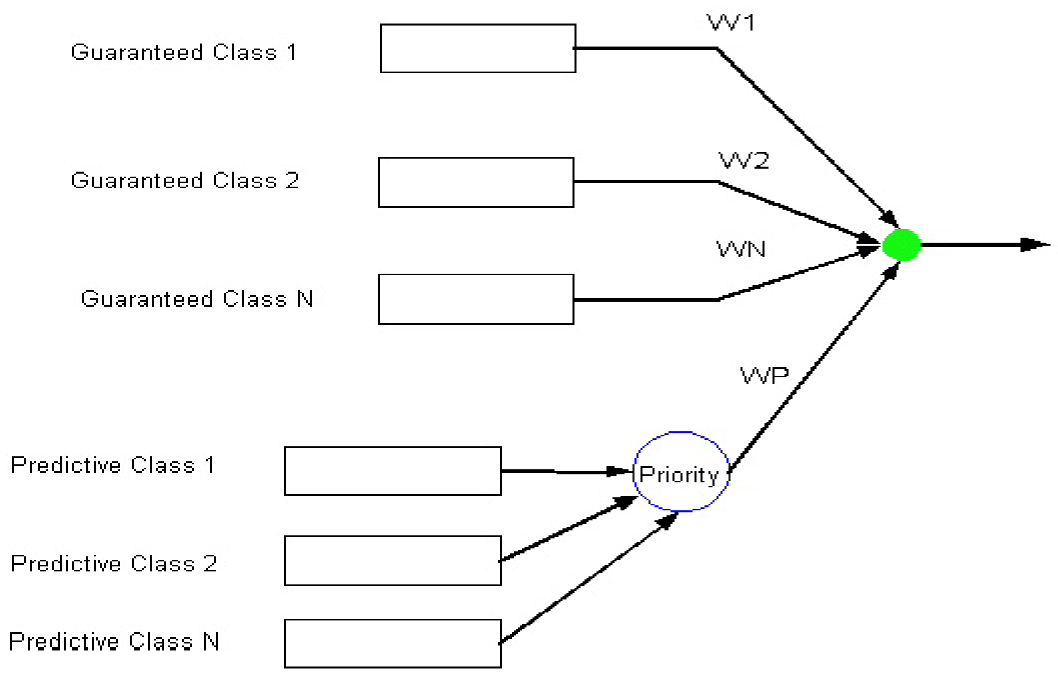


Figure 22: CSZ scheduler

3.3.1.11 Deficit Round Robin

Deficit Round Robin scheduler alleviates the problem with the various sizes of packets. The regular round robin is ignoring the fact that packets have different sizes and this causes some issues of fairness. DRR uses stochastic fair queuing to assign packets into the queues [4]. The queues are served with round robin with the only difference if a queue was not able to send a packet in the previous round because its packets was too large, the remainder from the previous quantum is added to the quantum for the next round. See Figure 23 and Figure 24.

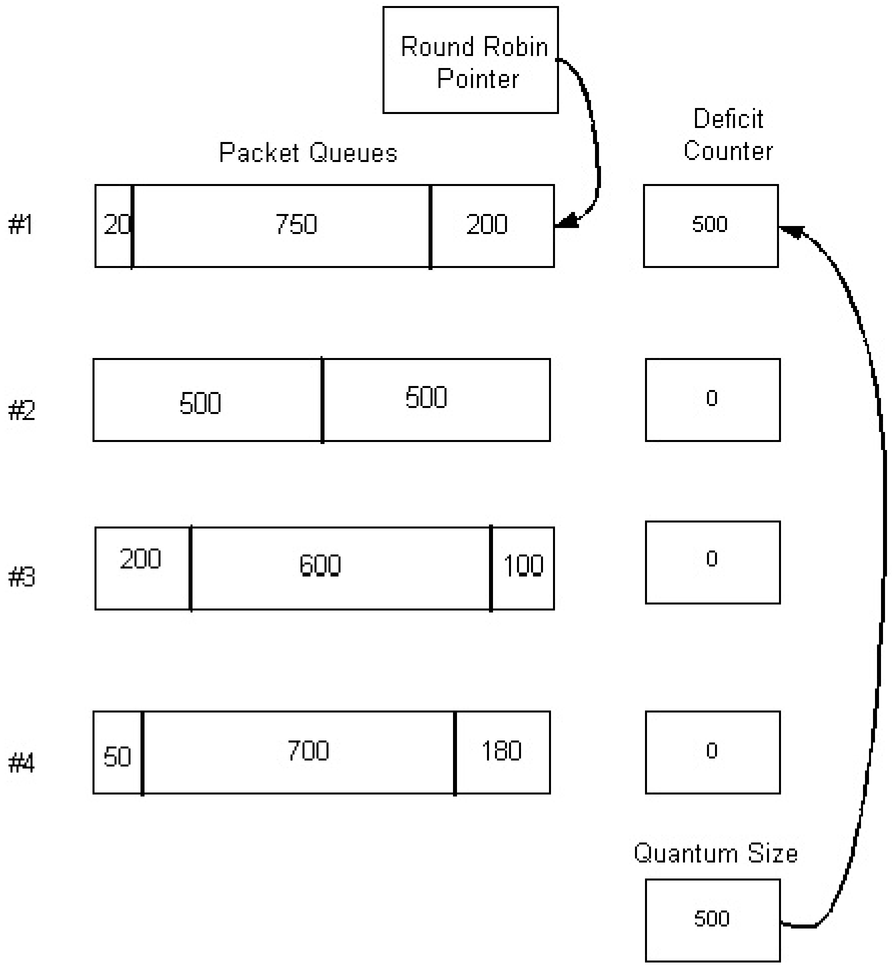


Figure 23: Deficit Round Robin: Initialize the variables to zero.

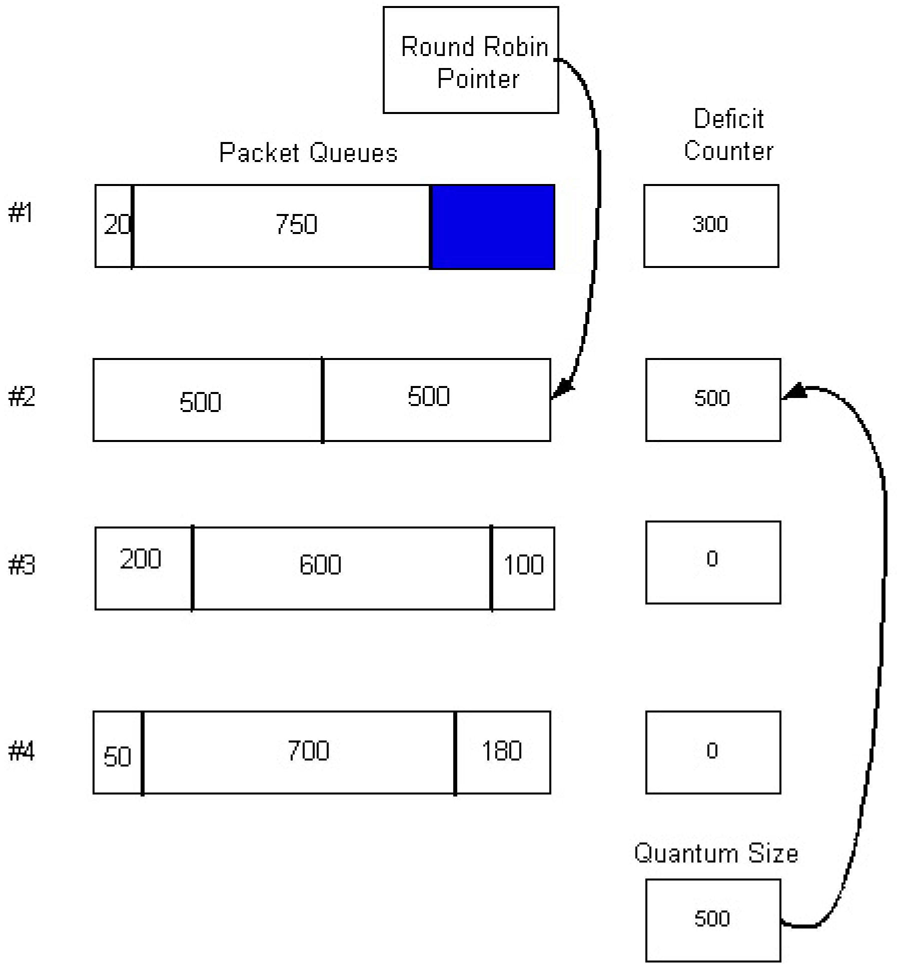


Figure 24: Deficit Round Robin: After sending out a packet of size 200, queue had 300 bytes of its quantum left.

3.3.1.12 Token Bucket Filter

The TBF is a simple queue that monitors the traffic that is transmitted by single source and limits the traffic on the desirable rate. Figure 25 shows the function of TBF. The bucket size, **b**, of the TBF is the most important parameter since it defines the numbers of tokens that can be stored. A token is removed from the bucket every incoming byte that is sent by the source. New tokens are placed back to the bucket based on the rate, **r**, of the token. When the bucket is empty the arriving packets are dropped.

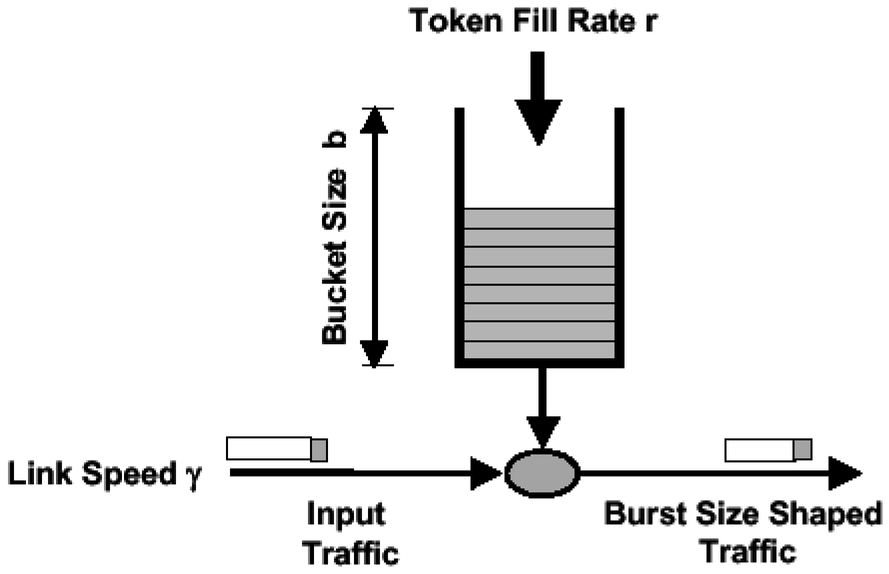


Figure 25: Token Bucket Filter

There are three possibilities based on the TBF algorithm:

- The data arrives into TBF at rate *equal* the rate of incoming tokens. In this case each incoming packet has its matching token and passes the queue without delay.

- The data arrives into TBF at rate *smaller* than the token rate. Only some tokens are deleted at output of each data packet sent out the queue, so the tokens accumulate, up to the bucket size. The saved tokens can be then used to send data over the token rate, if short data burst occurs.

- The data arrives into TBF at rate *bigger* than the token rate. In this case filter overrun occurs -- incoming data can be only sent out without loss until all accumulated tokens are used. After that, over limit packets are dropped.

4 Implementing Differentiated Services on Linux

There are many networking function that can be used to implement and deliver differentiated service. In this thesis we will briefly mention two software tools designed to configure a DiffServ router. One of the tools is known as UCLA software and the other the EPFL software.

The designers of the UCLA package wanted to have low level forwarding path for providing different levels of network services. A Bandwidth Broker manages the allocation of resources. The BB configures routers forwarding parameters in the domain accordingly with a policy database. The policy database stores information about flows requiring increased network service.
The UCAL package runs on FreeBSD Unix.

The EPFL software runs under Linux. Based on the modularity that Linux offers makes it a very flexible platform for experiments with PHBs already under standardization as well as experiments with new PHBs. And that's why we used the EPFL package for our experiments.

4.1 Introduction to Linux Networking Services

Linux is an open source operating system, which is freely available to the public. Linux had gained popularity all over the world but mostly in the academic environment. Most of the testbeds are released in Linux or in Unix environment first. Linux offers a rich set of Traffic Control (TC) functions for networking.

Lists of possible network traffic control functions include:

- Throttle bandwidth for certain computers
- Throttle bandwidth to certain computers
- Fairness for bandwidth sharing
- Protect your network from DoS attacks
- Protect the Internet from your customer
- Multiplex several servers as one, load balancing, or enchanted availability
- Restrict access to your computers

- Limit access of your users to other hosts
- Do routing based on user id, MAC address, source IP address, port type of service, time of day or content.

The Linux kernel offers support for IntServ, DiffServ and QoS. Before we get into the details of traffic control configuration of Linux we have to understand how the TC works under Linux. In order to transmit data into the network we have to setup the network card, using appropriate driver software.

Two functions of the driver software are:

- The Linux Networking Code can request the network driver to send a packet on the physical network.

- The network driver can deliver packets that it has received on the physical network to the Linux Networking Code. The current architecture sends the data from the application to the networking driver, see Figure 26.

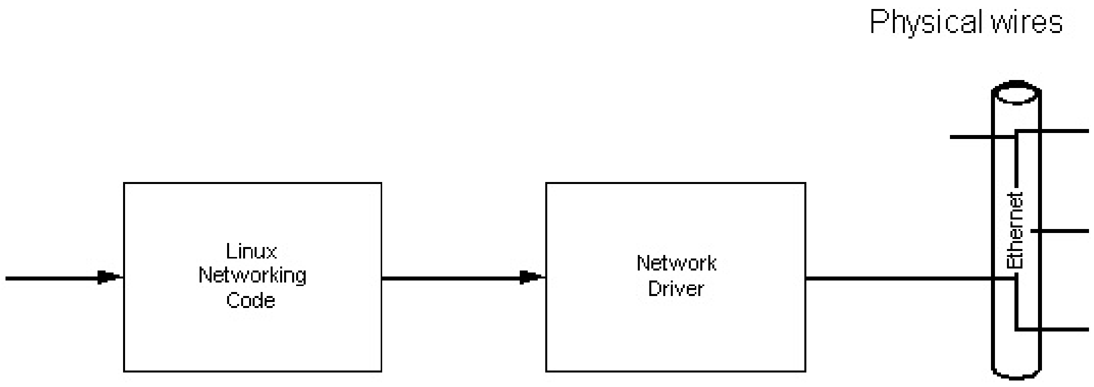


Figure 26: Default Setup of a Linux

Figure 27 shows an extra function, the Traffic Control function, included in the LINUX implementation. With the traffic control in between the Linux

Networking Code and the Network driver, packets can be manipulated in several ways.

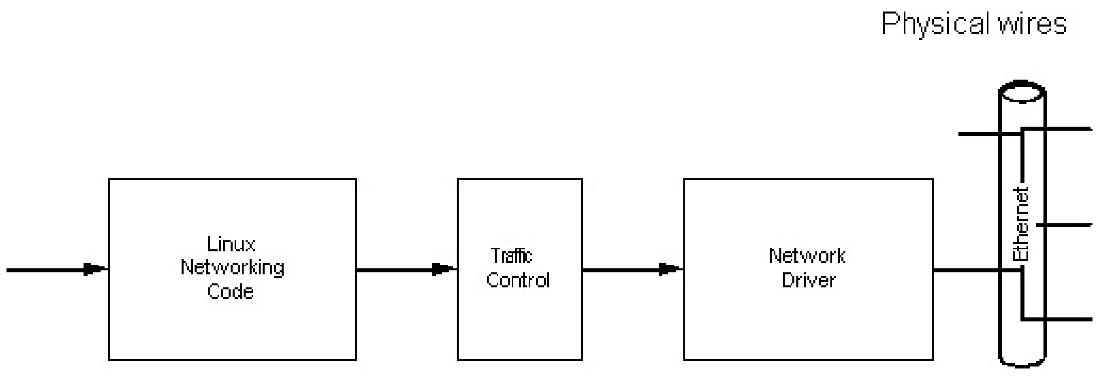


Figure 27: Linux Setup with Traffic Control

Figure 28 shows the block diagram of the kernel processes, the packets received from the network and how it generates new data to be sent on the network [5].

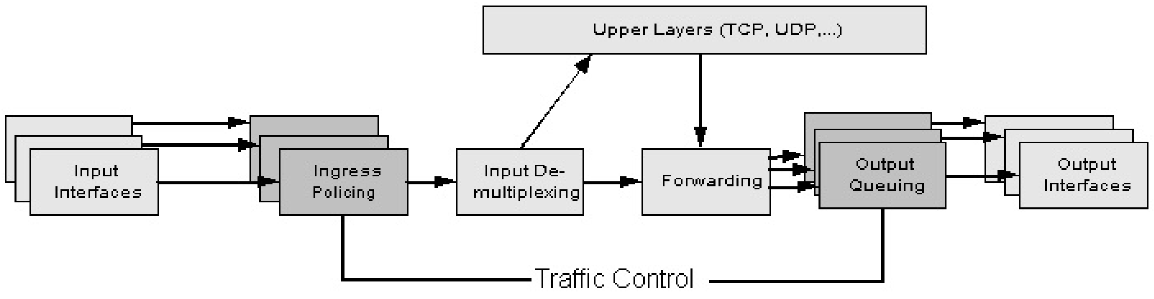


Figure 28: Processing of network data

The Input interface is responsible for passing packets to the Ingress Policing module. Packets could be policed at the Input interfaces. The Ingress policing modules are responsible for discarding traffic in the event that packets are arriving too fast. Then the packets are either forwarded on different interface, in case the machine is acting as a router, or are passed to the higher layers for further processing. The forwarding module is responsible for the selection of the output interface, the selection of the next hop, encapsulation etc. Then the packet is queued at the output interface. The traffic control can drop packets based on several parameters that can be selected by the user.

The major conceptual components of the traffic control of Linux code are:

- Queuing disciplines
- Classes (with queuing disciplines)
- Filters
- Policing (and related concepts)

4.2 Queuing Discipline

Every network device has its own queuing discipline that controls how the packets are enqueued. There are several queuing disciplines such as pFIFO, bFIFO, SFQ, RED and so on. Figure 29 shows a queuing discipline.

Queuing discipline

Figure 29: A simple queuing discipline without classes.

Figure 30 shows a queuing discipline that uses filters to prioritize packets into to different classes. More than one filter can be mapped to a class [7].

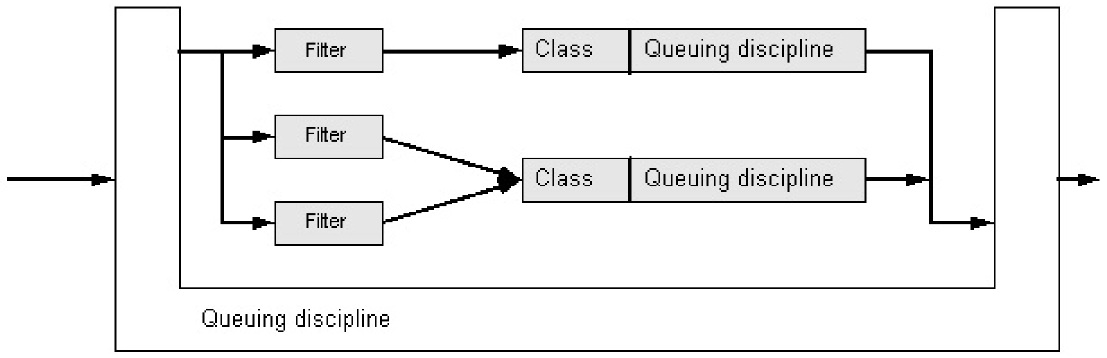


Figure 30:A simple queuing discipline with multiple classes.

Classes use another queuing discipline to store their data; such queuing disciplines can be pFIFO, bFIFO, RED, SFQ and etc. Figure 31 shows this scenario.

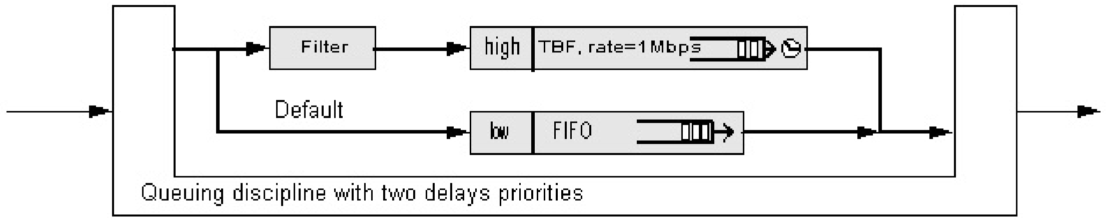


Figure 31: Combination of priority. TBF and FIFO queuing disciplines.

The queuing discipline implements a two-delay priority. The packets are filtered and classified into these two classes. The first queue is a token bucket filter, which is the high priority class. The TBF is served with 1Mbps rate and has a higher priority than the FIFO. All the other packets are classified into the lower priority queue, which is served with a queuing discipline FIFO.

Each queuing discipline is identified by unsigned 32 bit numbers, u32. The identification number of the queuing discipline is split into two parts, the major number and the minor number. The major number and the minor number are 16 bit each. The notation is major:minor, where a minor number is always zero for the queuing disciplines see Figure 32. At the network device eth0, there must be only one queuing discipline, which the major number of the queuing discipline must be unique. In case the user doesn't define the major number of the queuing discipline the system assigns one automatically.

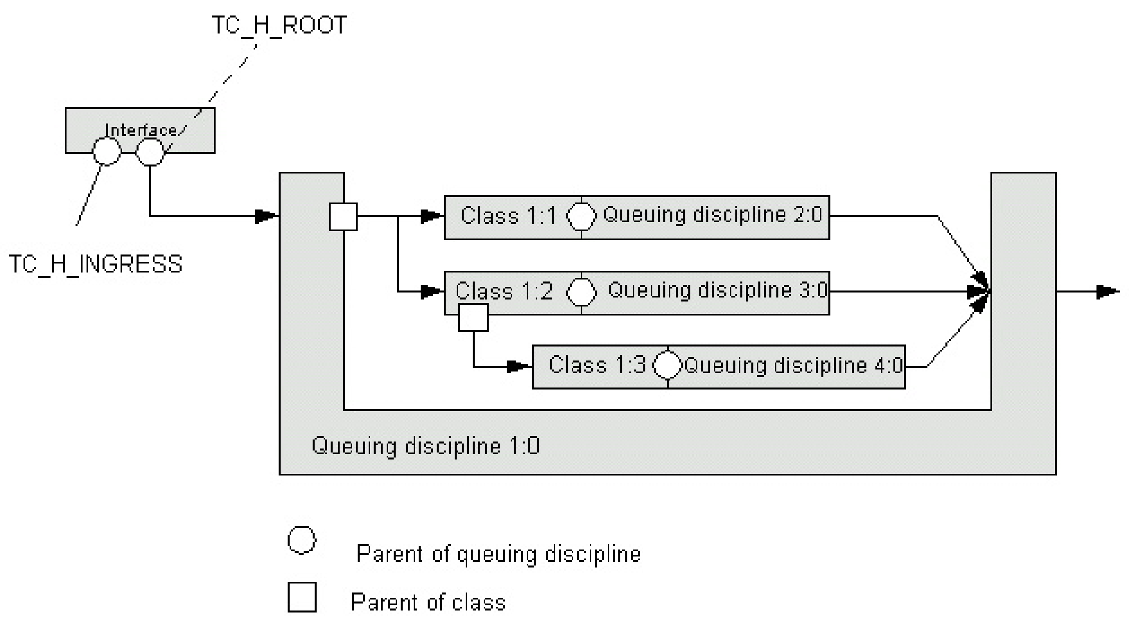


Figure 32: Addressing for queuing disciplines and classes.

Each queuing discipline has a set of certain functions that uses to control its operations. Such functions are enqueue, dequeue, requeue, drop, init, reset, destroy, change, and dump. More detail description of these functions can be found at [19]. There are some statistics that are maintained by each queuing discipline. The minimum sets of statistics that are maintained are the following:

- The current queue length
- The cumulative number of bytes enqueued
- The cumulative number of packets enqueued
- The cumulative number of packets dropped

4.3 Classes

There are two ways that you can identify a class, by the class ID, and the internal ID. The class ID is been assigned by the user, and the internal ID by the queuing discipline. The internal ID must be unique with a given queuing discipline. The data type of the Class ID is u32 and the internal ID is unsigned long. The kernel is accessing the class by its internal ID.

Queuing disciplines with classes provide a set of functions to manipulate classes. A list of these functions is graft, leaf, get, put, change, delete, walk, tcf_chain, bind_tcf, unbind_tcf and dump_class [19].

4.4 Filters

The incoming packets have to be assigned into the various classes. This is done using filters. Queuing disciplines are responsible for this task and with the usage of filters can assign incoming packets into classes. This takes place during the enqueue operation. The filters are organized based on the queuing discipline or class. All filters are stored in a filter list. This list is organized either by the queuing discipline or by class. It's also ordered based on the priority, in ascending order. The structure of the filters can been seen at Figure 33.

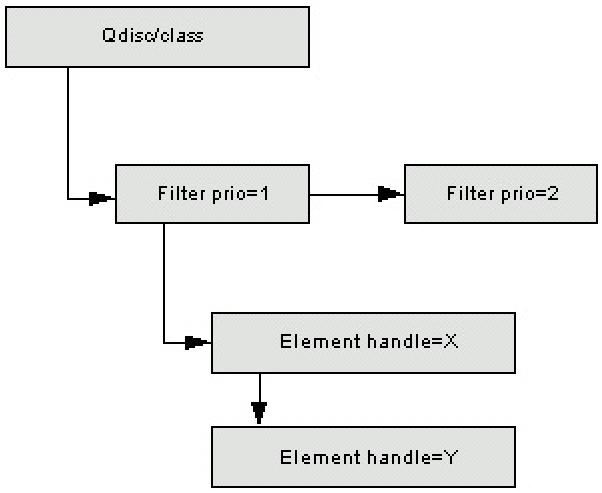


Figure 33: Structure of filters

Like classes, filters have internal IDs that are used to be reference for some internal tasks. Figure 33 shows the handles and the internal ID that are used for internal purposes. These handles are 32-bit and the internal IDs are unsigned long type. The order of which the filters and their elements are examined to get a match for the incoming IP is shown at Figure 34.

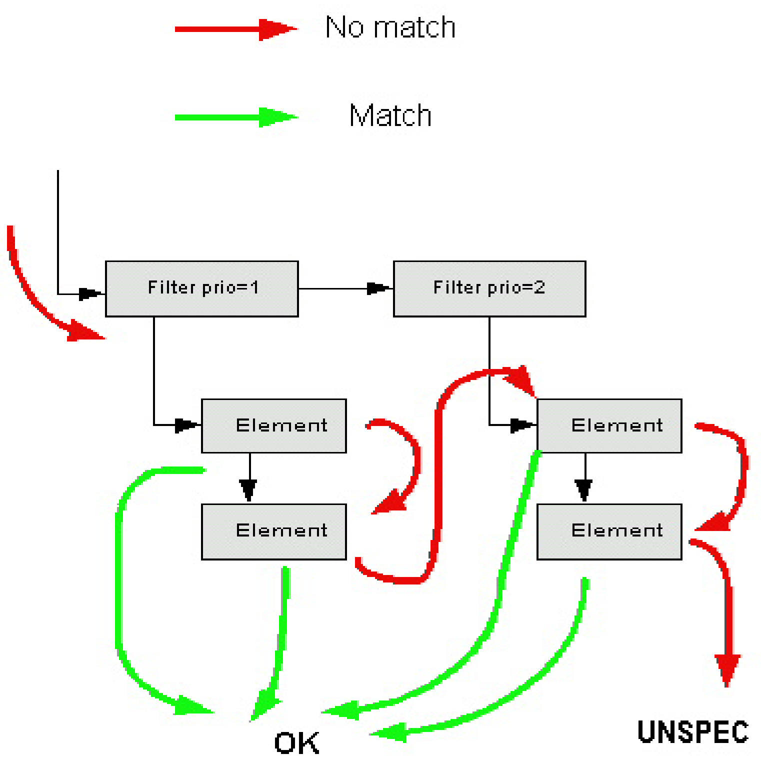


Figure 34: Looking for Filter Matching

There are several functions that can be used in order to control the filters. A list of these functions is classify, init, destroy, get, put, change, delete, walk, and dump. For more information regarding these functions can be obtained from [19] [5].

Filters are broken down to generic and specific filters. Generic filters can use one instance per queuing discipline that can classify packets for all classes. The cls_fw, cls_route, and cls_tcindex are generic filters. Specific filters use one or more instances of the filter or its internal element per class. The cls_rsvp and cls_u32 are specific filters.

4.5 Policing

In order to make sure that the traffic doesn't violate a certain limitation, we use policing. The policing is broken down to five policing mechanisms, policing decisions by filters, policing at ingress, refusal to enqueue packets, dropping packets from an inner queuing discipline and dropping a packet when enqeueing.

4.6 Classifiers under Linux

Some of the classifiers that are used by the **tc** program are the following:

- ***fw***
  Bases the decision on how the firewall has marked the packet.
- ***f32***
  Bases the decision on fields within packet (source-destination address, etc)
- ***route***
  Bases the decision on which route the packet will be routed.
- ***rsvp, rsvp6***
  Bases the decision on the target (destination, protocol) and optionally the source as well.

The classifiers that we have listed above accepted several parameters that some of them are common. A list of these parameters follows:

- ***protocol***
  The classier defines the protocol that will accept. Required IP only.
- ***parent***
  The handle this classifier is to be attached to. This handle must be an already existing class. Required.
- ***prio***
  Defines the priority of this classifier.
- ***handle***
  This handle means different things to different filters.

5 Evaluating Differentiated Services on Linux

5.1 Topology Under Study

Figure 35 shows the network setup that we have implemented at University of Cyprus (UCY). The Linux router is an Acer PII300MHZ, 128MRAM. We have installed Linux Mandrake with a Kernel 2.4. The clients are connected on a 100Mbps switch and the outgoing interface of the router goes on a 10Mbps Hub.

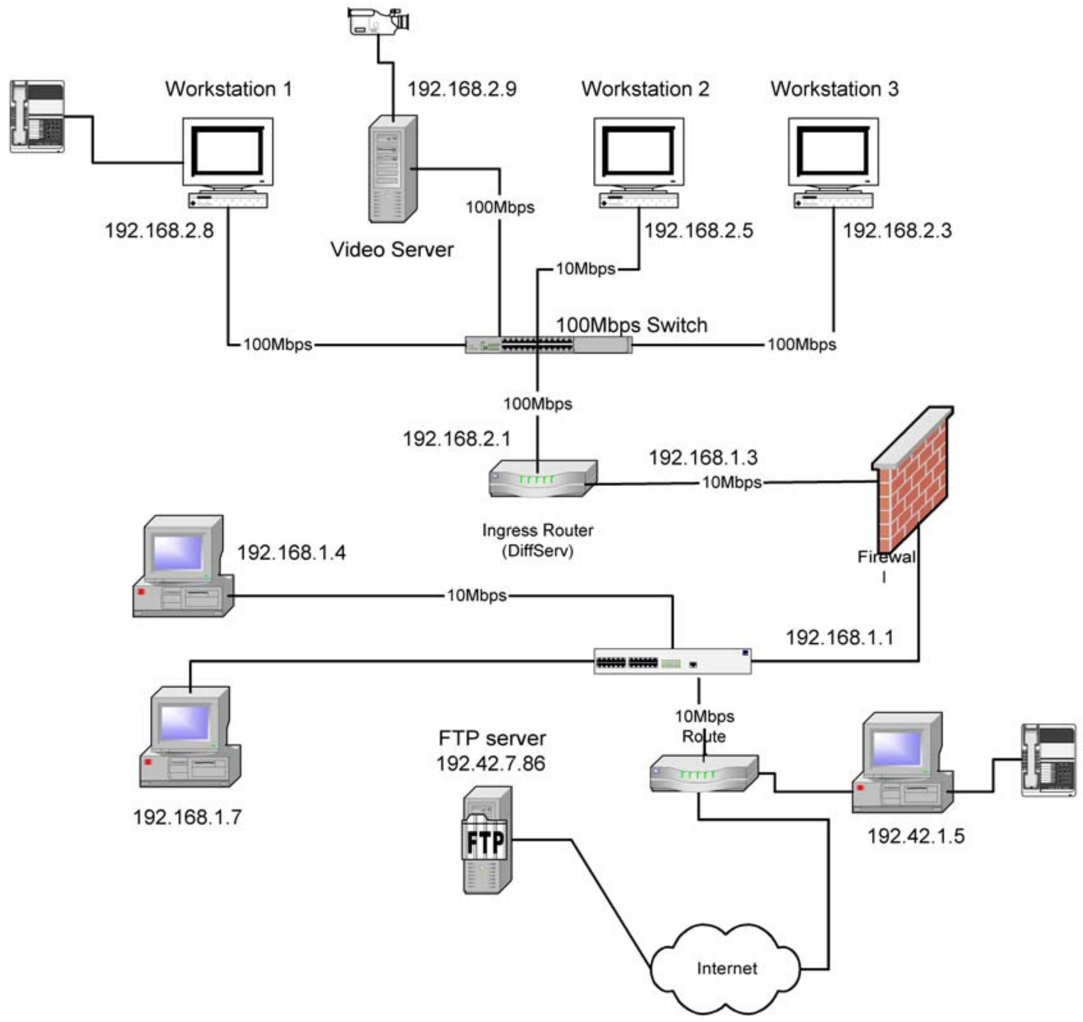


Figure 35: Network Topology for DiffServ Architecture

In order to obtain some statistics we have used various tools such as IPERF, TCPDUMP, TCPTRACE and XPLOT. IPERF generates UDP and TCP data traffic. It has the ability to transmit the data at specific port, or at specific bit rate, or a certain amount of data. IPERF runs under Linux and Windows. When we used IPERF under windows we have noticed some discrepancies. Another tool that we have used is the TCPDUMP. TCPDUMP captures the traffic at the Ethernet card. And last, we have used the TCPTRACE tool. TCPTRACE analyses the data that are generated from the TCPDUMP. TCPTRACE generates some files that can be used by the XPLOT tool to generate graphs.

5.2 Evaluating Linux Implementation for DiffServ

This section focuses on the evaluation of the DiffServ architecture under Linux. Several Scenarios were considered which aim to show the behavior of the Linux implementation of Differentiated Services under various queuing disciplines, topologies, various parameters etc.

We have broken down the results in two categories. The first category is the scenarios that have one queue. For this category we have run tests to verify the expected behavior of the queuing disciplines for non-DiffServ networks. The results can be found in Appendix I.

The second category is the scenarios with two queues and with DiffServ implementation. The results appear in section 5.2.1. First we present a table giving an overview of the four scenarios we have investigated.

SCENARIO 1						
	Scheduler	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type
Test 1						
	CBQ	5	500K	1	pFifo	UDP
		5	500K	1	pFifo	UDP
Test 2						
				100		
				1		
Test 3						
		1		1		
		5		1		
Test 4						
		4		1		
		5		1		
Test 5						
		5	800K	1		
		5	200K	1		
Test 6						
		5	500K	1		UDP
		5	500K	1		TCP
Test 7						
		5	500K	1		TCP
		5	500K	1		TCP

Table 2

Scenario 2						
Test 1						
	Scheduler	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type
	CBQ	5	200K	2	TBF	UDP
		5	800K	1	pFifo	UDP
Test 2						
						UDP
						TCP

Table 3

Scenario 3						
	Scheduler	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type
Test 1						
	PQ	PQ1	200K	5	pFifo	UDP
		PQ2	800K	5	pFifo	UDP
Test 2						
	PQ	PQ2	200K	5	pFifo	UDP
		PQ1	800K	5	pFifo	UDP

Table 4

Note: unfilled table entries indicate that they use the same parameters as in test 1.

Scenario 4						
	Scheduler	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type
Test 1						
	CBQ	5	300K	2	pFifo	UDP
		5	700K	1	RED	UDP
Test 2						
						TCP
						TCP
Test 3						
						UDP
						TCP
Test 4(TCP Window size 64K)						
			500K			TCP
			500K			TCP
Test 5 (TCP Window size 128K)						
			500K			TCP
			500K			TCP

Table 5

Note: unfilled table entries indicate that they use the same parameters as in test 1.

5.2.1 Scenario 1 (pFIFO, pFIFO)

Test 1

In this scenario we are using two pFifo queues, see Figure 36.The source 192.168.2.5 generates traffic for the receiver 192.168.1.4. The flow 1 represents the traffic that travels from 192.168.2.5 through the upper pFifo and to the 192.168.1.4. The flow 2 represents the traffic that starts from 192.168.2.3 and traverse through the lower pFifo towards to 192.168.1.7. From the Table 6, we can see that both of the flows have the same priority and the same weight. Both of the sources transmit the same amount of data at the same bit rate. In all the tests, we have weighted the super class of the CBQ at 1Mbit.

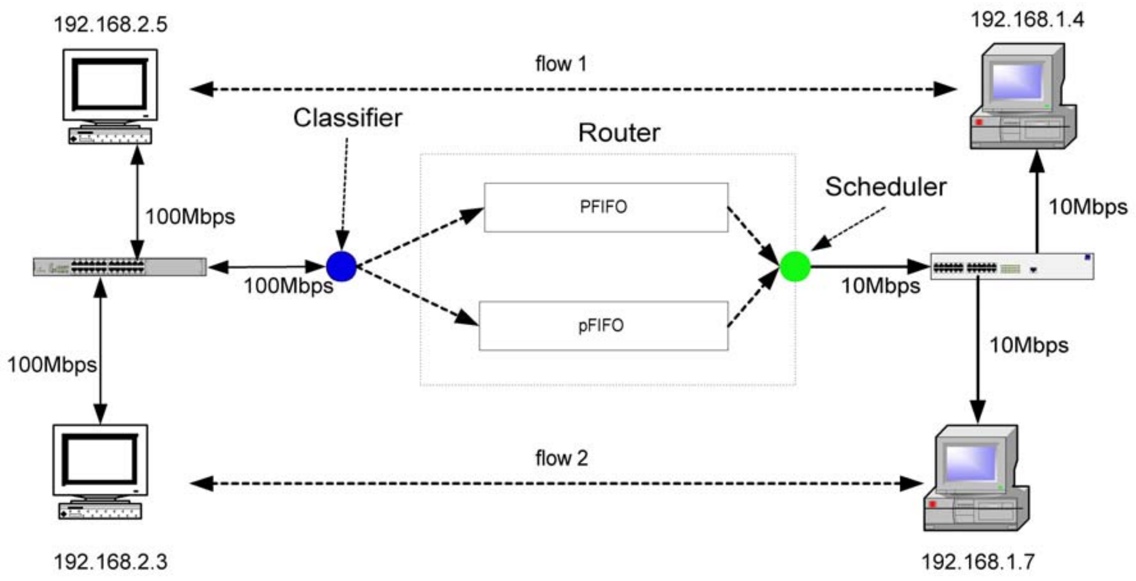


Figure 36: Block Diagram of pFifo queues
with same class and filter priorities and weights

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	500K	1	5Mbps	
192.168.2.3	U32	CBQ	5	500K	1	5Mbps	

Table 6

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	pFifo	50	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	pFifo	50	UDP	14267	22.9

Table 7

Flow 1 and flow 2 are 5Mbps each. Table 6 shows the transferred data and the bit rates that we have been transmitting from the sources.

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	4.5Mbps	13.1Mbytes	9.294	4926	14267	23.1
192.168.1.7	4.6Mbps	13.1Mbytes	1.421	4918	14267	23

Table 8

The results from this test can be seen in Table 7 and Table 8. The results appear reasonable since the incoming rate at the receivers is below 5Mbps. This test shows fairness over the UDP flows.

Test 2

In this test, we would like to investigate the effect of the filter priorities. Table 9 shows the setting of the filter priorities.

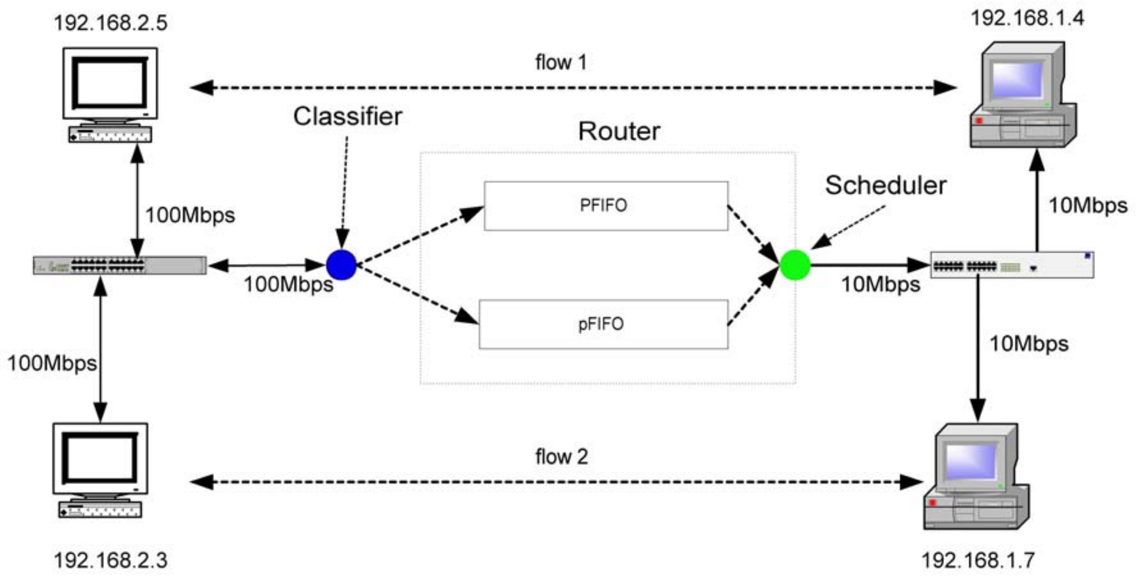


Figure 37: Block Diagram of pFifo queues with different filter priorities

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	500K	100	5Mbps	
192.168.2.3	U32	CBQ	5	500K	1	5Mbps	

Table 9

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	pFifo	50	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	PFifo	50	UDP	14267	22.9

Table 10

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	4.5Mbps	13.0Mbytes	8.294	5029	14267	23.1
192.168.1.7	4.6Mbps	13.3Mbytes	1.345	4913	14267	23

Table 11

The results are similar to the test 1; see Table 10 and

Table 11. In this test, observe that the filter priorities do not play a critical role on the bandwidth allocation. We have to keep in mind that the filter priorities are for the classification of the data into the queues. Of course, we cannot assume the same if we were sending TCP and UDP traffic at the same time. An important observation is that the jitter of the two flows seems questionable. IPERF, the tool that we are using, uses the formula 1, to compute the jitter.

$$E\{(W_i)-E[W_i])]\} \qquad \text{Formula (1)}$$

Wi is a random delay that rises out of the buffering within network

After analyzing the formula we can notice that the results are correct. Keep in mind that was impossible to synchronize the two sources to start transmitting data at the same time. By knowing this, on of the two sources can take the advantage of not giving very accurate results. We noticed the following behavior; at the starting time the source that started transmitting first had a different jitter than the other. At the period that both of the sources were transmitting we noticed the same jitter at both ends, and towards the end we noticed again different jitter time since one of them had finished transmitting.

Test 3

In this test, we set the class priorities. Flow 1 has a higher priority over flow 2, see Table 12. All the other parameters remain the same as were in test 1.

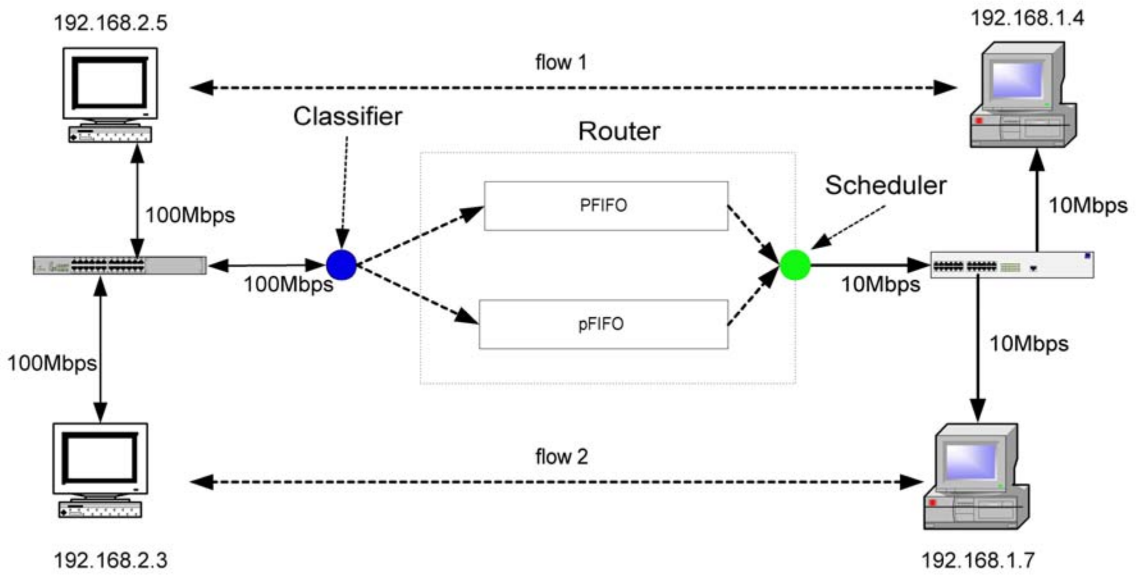


Figure 38: Block Diagram of pFifo queues with different classes priorities and same weights

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	1	500K	1	5Mbps	
192.168.2.3	U32	CBQ	5	500K	1	5Mbps	

Table 12

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	pFifo	50	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	pFifo	50	UDP	14267	22.9

Table 13

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	7.0Mbps	20.0Mbytes	0.708	0	14267	22.9
192.168.1.7	2.1Mbps	6.2Mbyte	11.459	9876	14267	23.1

Table 14

Table 13 and Table 14 show that we can differentiate flows with higher priorities. The results are remarkable since we got rates up to 7Mbps on the 192.168.1.4. The lower the number is set at the class priority, the higher the priority it has over the other flow. When a class has a higher priority over the other one, the scheduler has to execute the packets in that class and then move to the next one.

Test 4

In test 4 we investigate the sensitivity of the priority level. For instance, what is the relation between two classes that have priority 1 and 5 and 4 and 5.

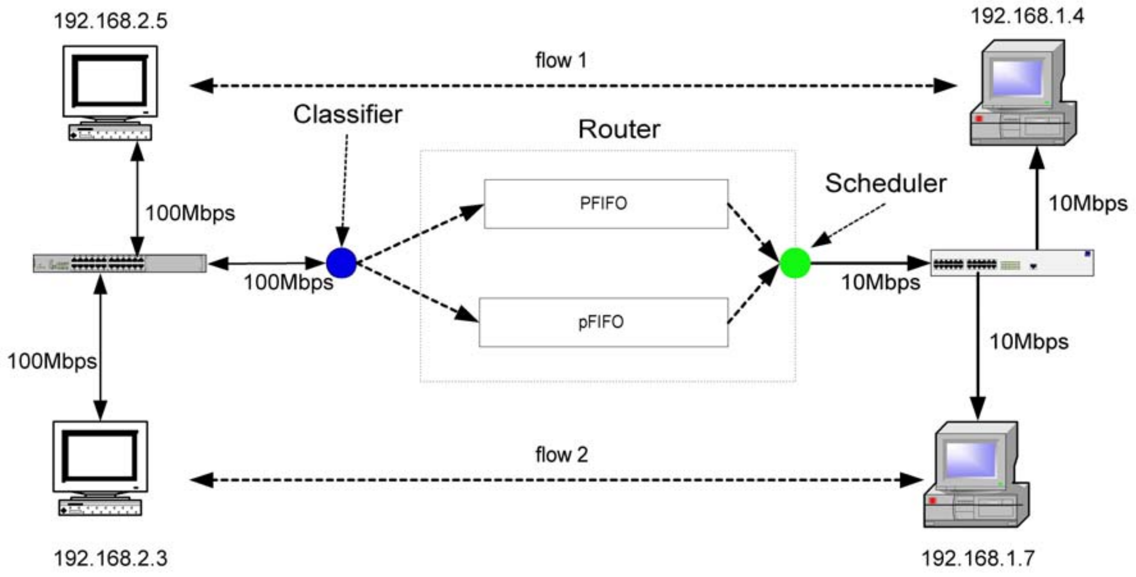


Figure 39: Block Diagram of pFifo queues
with same class and filter priorities and weights

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	4	500K	1	5Mbps	
192.168.2.3	U32	CBQ	5	500K	1	5Mbps	

Table 15

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	pFifo	50	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	pFifo	50	UDP	14267	22.9

Table 16

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	7.0Mbps	19.9Mbytes	0.683	47	14267	22.9
192.168.1.7	2.2Mbps	6.2Mbyte	12.690	9838	14267	23.1

Table 17

The results of this test show that there is not much of a differentiation among the distances of the priorities. As far we got a difference among the priorities is good enough in order to give a higher priority to the queue.

Test 5

Here we investigate the weight behavior of a class. In order to accomplish this, we change the weights of the classes to be non-proportional to their bandwidths.

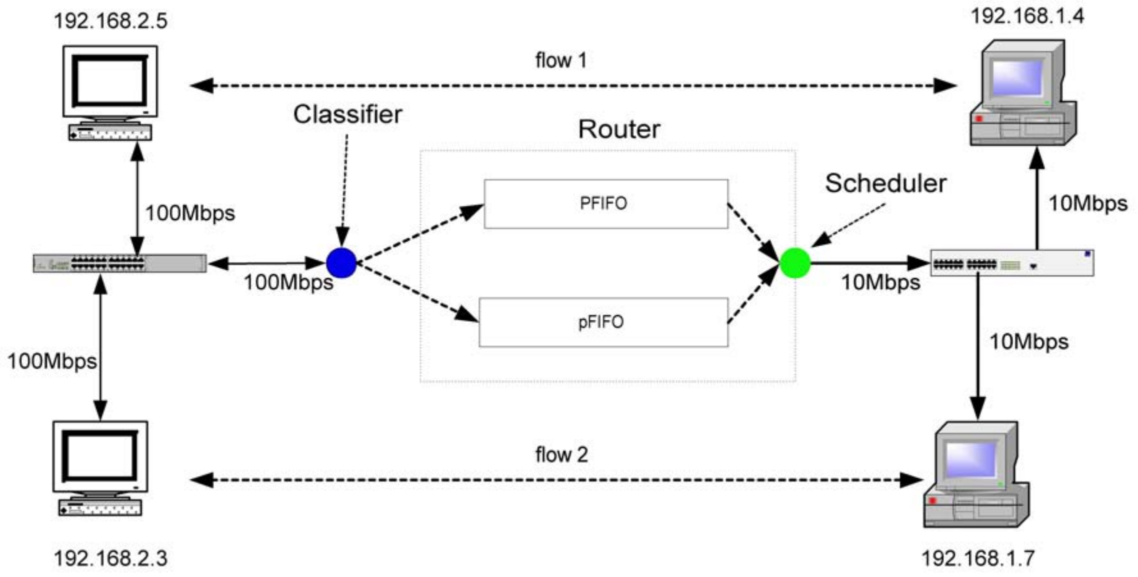


Figure 40: Block Diagram of pFifo queues
with same class priorities and filter priorities but different weights

Table 18 shows the parameters of the routers. The weights are 800K for flow 1 and 200K for flow 2.

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	800K	1	5Mbps	
192.168.2.3	U32	CBQ	5	200K	1	5Mbps	

Table 18

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	pFifo	50	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	PFifo	50	UDP	14267	22.9

Table 19

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	6.6Mbps	19.0Mbytes	0.683	691	14267	22.9
192.168.1.7	2.5Mbps	7.2Mbyte	8.838	9154	14267	23.1

Table 20

Table 19 and Table 20 show the results of this test. The results confirm the expectation, since the incoming traffic at the receivers is adjusted based on the weight value. This behavior is expected since the weights take place when the class priorities are the same.

Test 6

In this test, we change the traffic type that we are generating at the sources. The source 192.168.2.5 generates UDP traffic and the 192.168.2.3 generates TCP traffic.

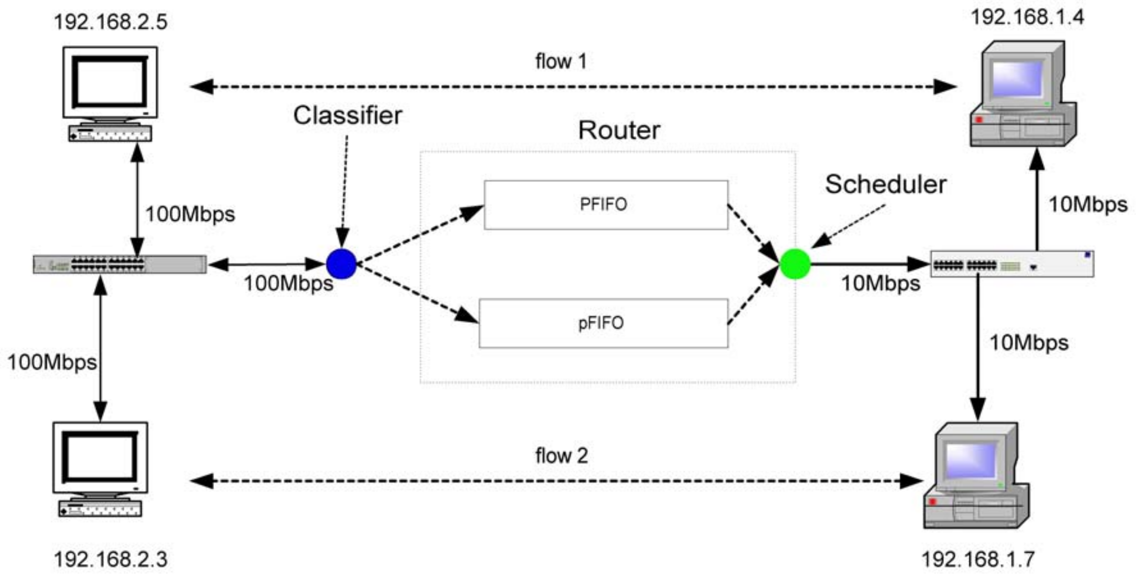


Figure 41: Block Diagram of pFifo queues with the same class priorities and weights

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	500K	1	5Mbps	
192.168.2.3	U32	CBQ	5	500K	2	5Mbps	

Table 21

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	PFifo	50	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	3.8	1514	PFifo	50	TCP	14267	41.7

Table 22

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	6.9Mbps	19.9Mbytes	0.736	100	14267	22.9
192.168.1.7	3.8Mbps	20Mbytes	----------	0		41.7

Table 23

It's obvious that UDP is getting most of the bandwidth at 6.9Mbps and the TCP is getting 3.0Mbps. This behavior is as expected, since the UDP traffic rate is not controlled. In contrast TCP is flow controlled, using a variant of the Jacobson algorithm. Observe that the uncontrolled UDP behavior has a substantial number of losses (recall no flow control, no sensing for retransmission of lost packets)

Test 7

Test 7 sets both sources to use TCP.

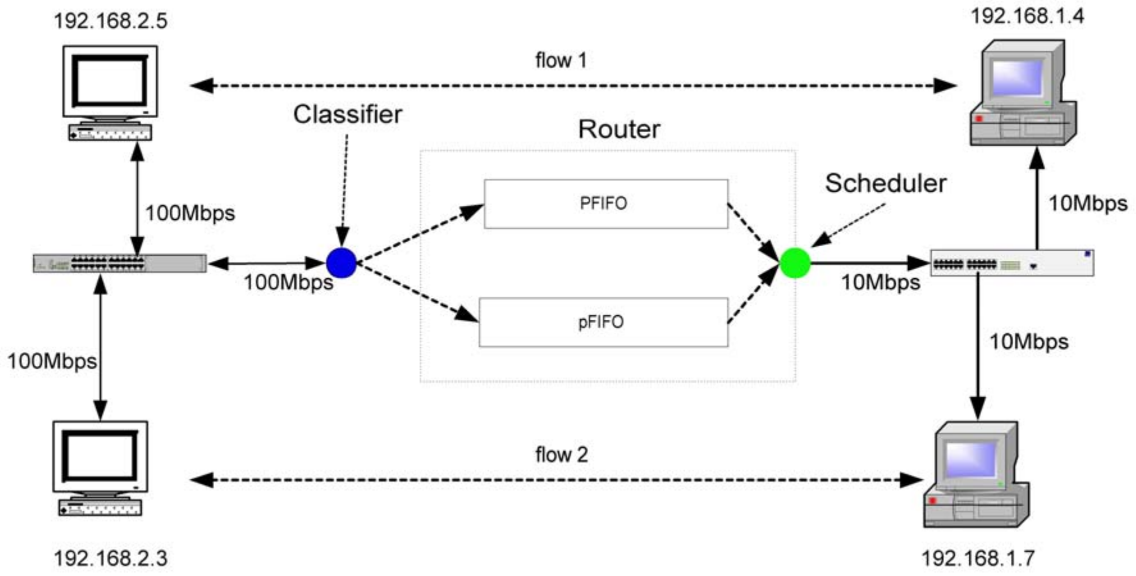


Figure 42: Block Diagram of pFifo queues with the same class priorities and weights

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	500K	1	5Mbps	
192.168.2.3	U32	CBQ	5	500K	2	5Mbps	

Table 24

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	3.5	1514	pFifo	50	TCP		46.5
192.168.2.3	192.168.1.7	20	3.4	1514	PFifo	50	TCP		47.5

Table 25

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	3.5Mbps	20Mbytes	----------			46.5
192.168.1.7	3.4Mbps	20Mbytes	-----------			47.5

Table 26

We can see from Table 25 and Table 26 that using TCP in both flows we get a fair treatment over the packets. We have to note that in TCP we mostly get zero packets losses, since the TCP window size is not big enough to exceed the bandwidth rate. At later a stage, we will show the effect of increasing the TCP window size.

5.2.2 Scenario 2 (TBF, pFIFO)

In this scenario, we change queuing disciplines. Here we use a Token Bucket Filter, TBF, at the upper queue and pFIFO at the lower queue. Also, we have allocated the bandwidth differently from the previous scenario. Here we give 2Mbps to the upper queue of the TBF queue and 8Mbps to the pFifo. Even though we have allocated 2Mbps to the TBF class, the TBF have been configured to limit the traffic at 1.5Mbps.

Test 1

In this test we have set up both of the classes with the same priority. The weights have been set based on the bandwidth allocation. The flow 1 gets 1.5Mbps and flow 2 gets 8Mbps. Both of the sources generate UDP traffic at 7Mbps.

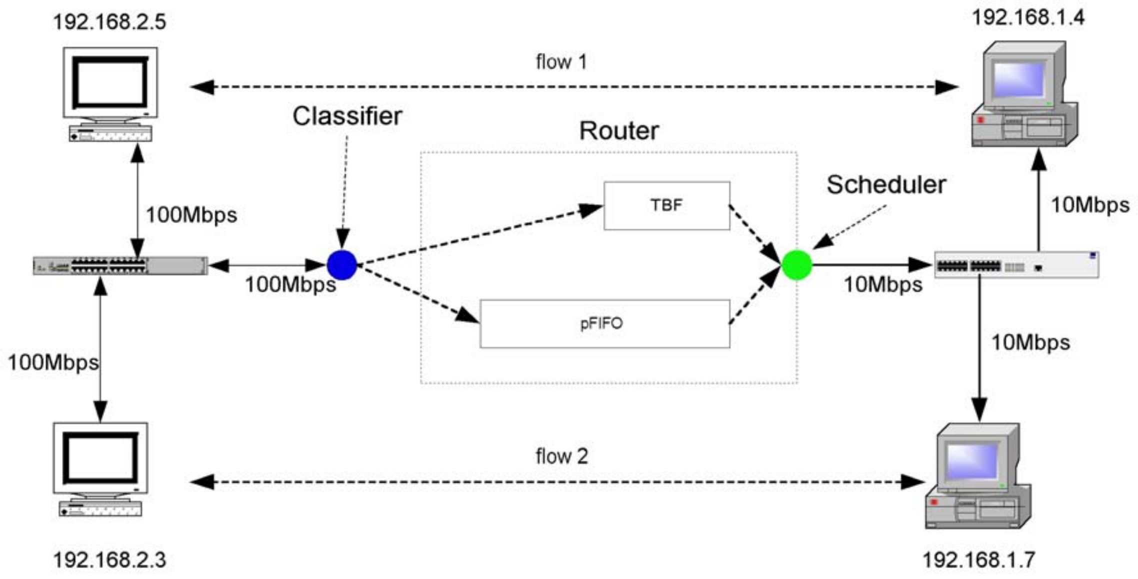


Figure 43: Block Diagram of TBF. and pFifo with same class and filter priorities and weights

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	200K	2	2Mbps	
192.168.2.3	U32	CBQ	5	800K	1	8Mbps	

Table 27

The parameters of the TBF are the following:

Rate 1.5Mbps

Burst 1.5KByte

Limit 1.5Kbytes

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	TBF		UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	pFifo	50	UDP	14267	22.9

Table 28

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	1.1Mbps	3.2Mbytes	15.313	11979	14267	23.1
192.168.1.7	7.0Mbps	20Mbyte	0.268	0	14267	22.9

Table 29

The results are outstanding. We can see that the incoming traffic at the receivers is almost what we have expected. Keep in mind that we have limited the upper queue, TBF, at 1.5Mbps. Even though, there are some looses at the UDP traffic we can ignore since the main goal here is to limit UDP traffic not to steal traffic from other classes.

Test 2

In this test we transmit UDP traffic through flow 1 and TCP through flow 2.

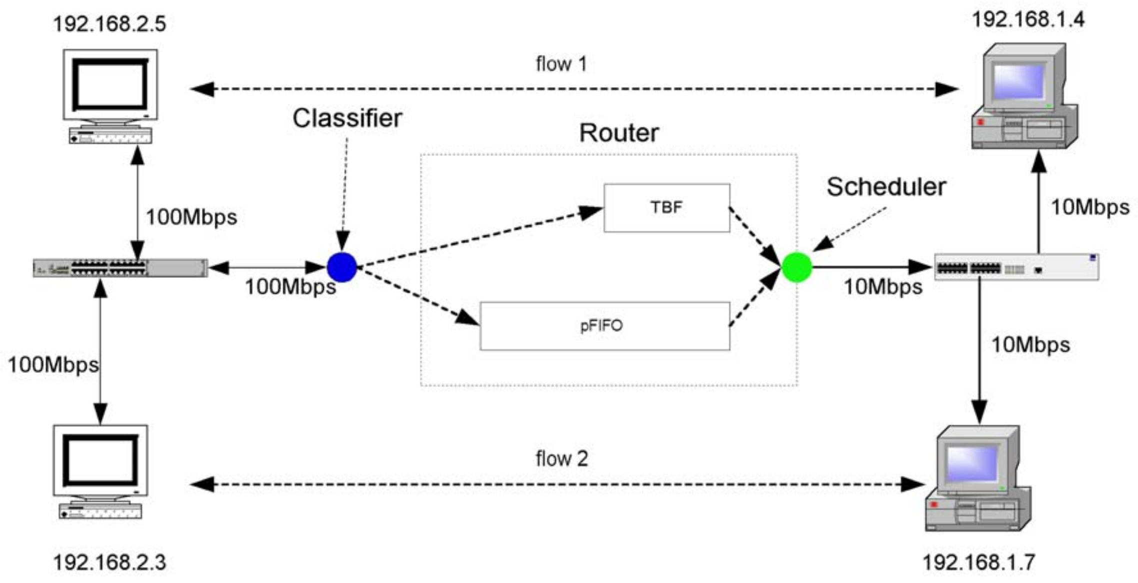


Figure 44: Block Diagram of EF PHB

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	200K	2	2Mbps	
192.168.2.3	U32	CBQ	5	800K	1	8Mbps	

Table 30

The parameters of the TBF are the following:

Rate 1.5Mbps

Burst 1.5KByte

Limit 1.5Kbytes

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	TBF		UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	pFifo	50	TCP	14267	28.0

Table 31

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	1 Mbps	2.8Mbytes	21.664	12255	14267	23.1
192.168.1.7	5.7Mbps	20Mbyte				28.1

Table 32

Table 32 and Table 33 show something very interesting. Using, TBF we can limit the UDP traffic from stealing traffic from other classes. Keep in mind that we have limited the upper queue, TBF, at 1.5Mbps. Even though, there are some looses at the UDP traffic we can ignore since the main goal here is to limit UDP traffic not to steal traffic from other classes. We can conclude that TBF is a good limiter for UDP traffic.

5.2.3 Scenario 3 (Priority Queues)

In this scenario we evaluate the Priority Queues. The main focus here is to see the behavior of the PQ. Figure 45 shows the topology.

Test 1

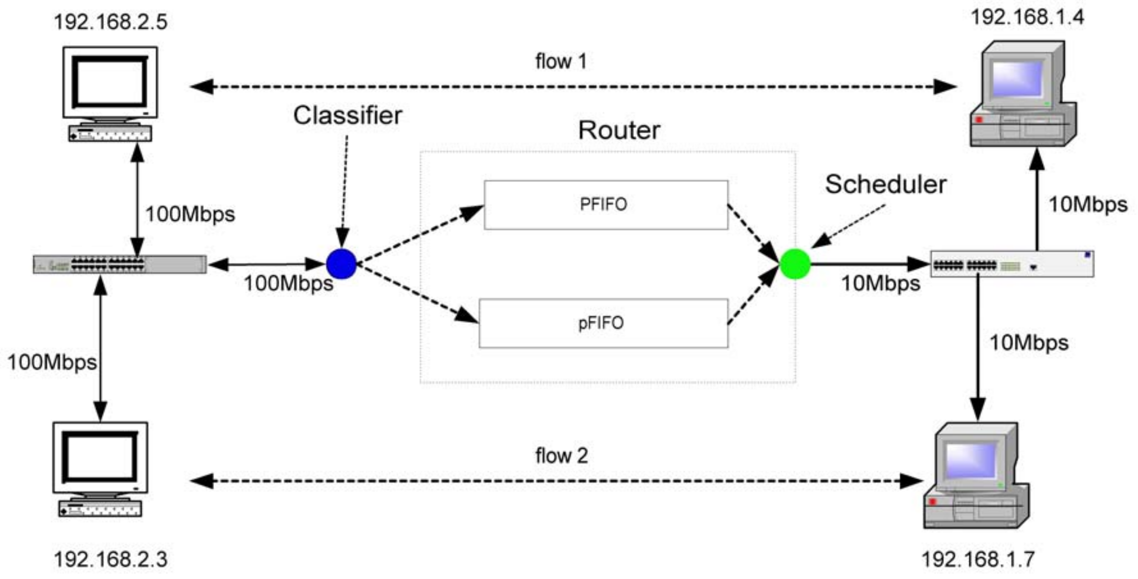


Figure 45: Block Diagram of Priority Queue

The PQ discipline is executing first the queue with the highest priority and then the rest. In our test we have assign flow 1 to be classified at Priority 1, which has the highest priority and flow 2 on the lower priority queue.

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	PQ	PQ1		5		
192.168.2.3	U32	PQ	PQ2		5		

Table 33

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514			UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514			UDP	14267	22.9

Table 34

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	7.0Mbps	20Mbytes	0.759	0	14267	22.9
192.168.1.7	2.2Mbps	6.2Mbyte	8.322	9829	14267	23.1

Table 35

As it expected flow 1 gets most of the bandwidth. This means that it has priority over the others. It’s obvious that flow 1 gets more priority than flow 2.

Test 2

In order to prove the consistency of this test we have reversed the priorities of the flows and we get the results that we expected. See the following tables 32,33 and 34

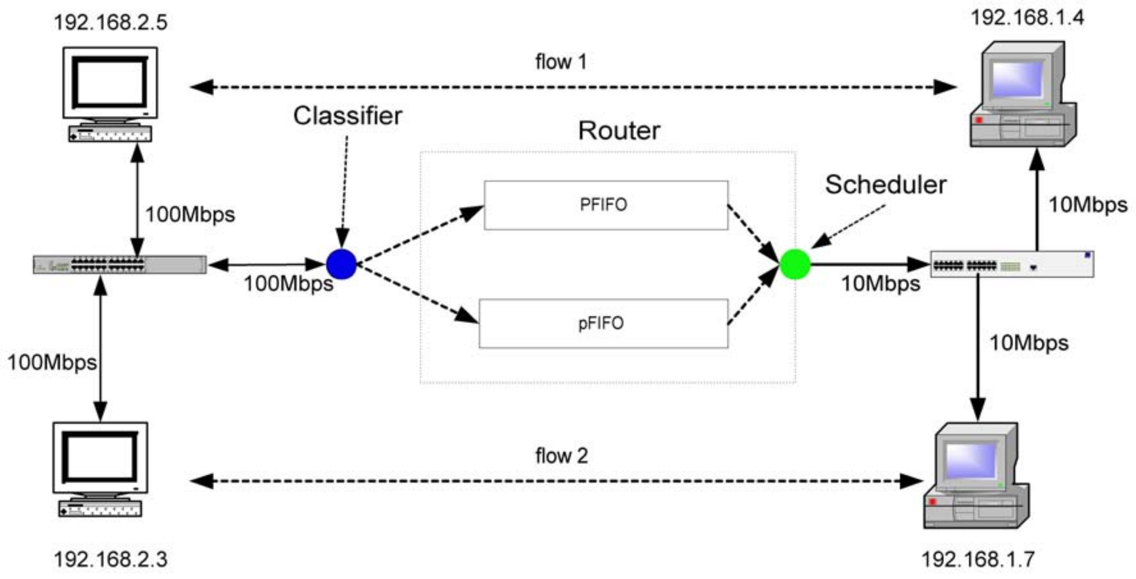


Figure 46: Block Diagram of Priority Queue

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	PQ	PQ2		5		
192.168.2.3	U32	PQ	PQ1		5		

Table 36

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514			UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514			UDP	14267	22.9

Table 37

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	2.2Mbps	6.2Mbytes	3.371	9839	14267	23.0
192.168.1.7	7.0Mbps	20Mbyte	0.735	0	14267	22.9

Table 38

Flow 1 is assigned to the priority queue 2. This means that it has lower priority over the others.

5.2.4 Scenario 4 (pFifo, RED)

DiffServ architecture has been focusing on various Per Hop Behavior groups. One of the most important one is the Expedited Forwarding. In this scenario, we have implemented an EF PHB, and we have analyzed it to a certain extent.

Test 1

In test 1 of the 3rd scenario we have two queues; a pFifo (upper queue) and a RED (lower queue). In this test the priorities of the classes and the filters are set the same. The weights of the classes are proportional to the bandwidth of the classes. Flow 1 has 3Mbps and flow 2 has 7Mbps. Both sources transmit UDP traffic at 7Mbps.

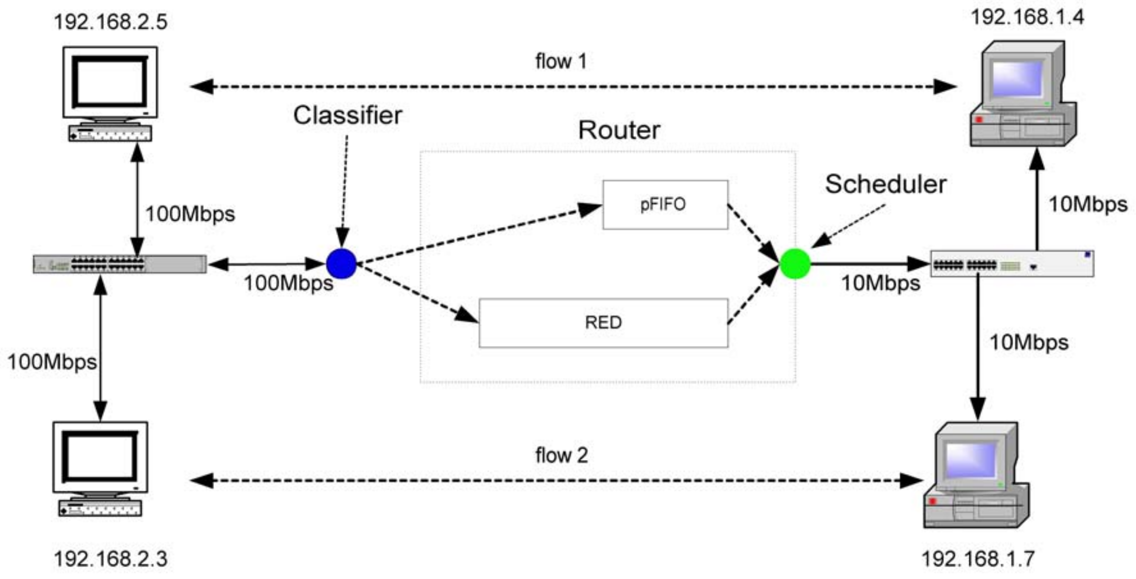


Figure 47: Block Diagram of EF PHB

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	300K	1	3Mbps	
192.168.2.3	U32	CBQ	5	700K	2	7Mbps	

Table 39

The parameters for the RED queue are the following:

Limit 60KB

Maximum 45KB

Minimum 15KB

Probability 0.1

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	pfifo	10	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	7	1514	RED		UDP	14267	22.9

Table 40

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	2.7Mbps	7.9Mbytes	0.708	8659	14267	23.1
192.168.1.7	6.4Mbps	18.2Mbyte	0.596	1253	14267	22.9

Table 41

The results are matched expectations. We get 2.7Mbps and 6.4Mbps for flow 1 and flow 2 respectively. Flow 1 is been limited at 3Mbps and flow 2 at 7Mbps.

Test 2

In this test we change the traffic type of the sources. Both sources transmit TCP traffic.

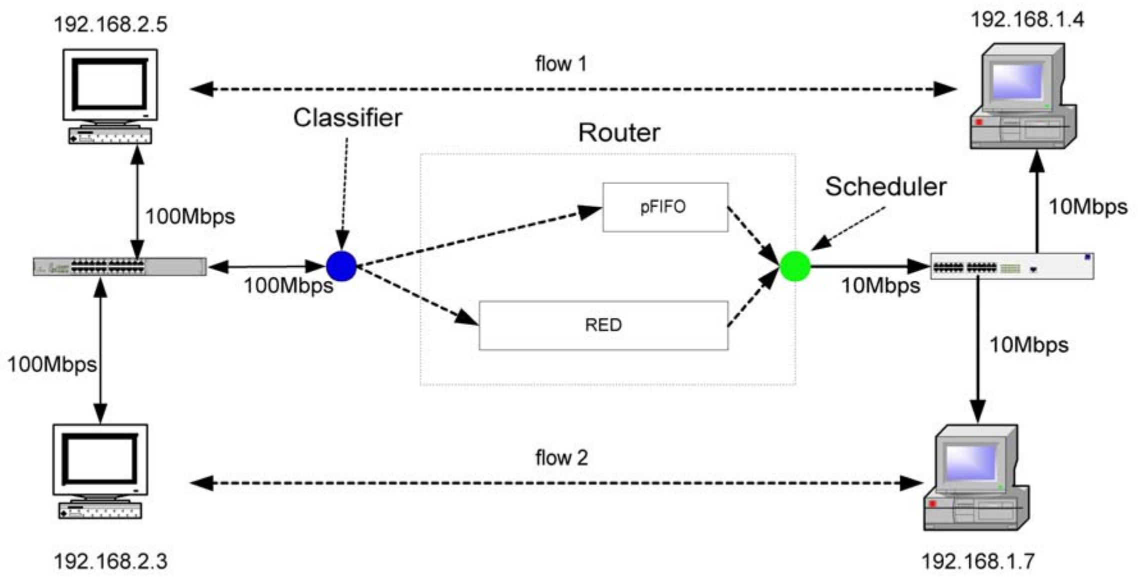


Figure 48: Block Diagram of EF PHB

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	300K	1	3Mbps	
192.168.2.3	U32	CBQ	5	700K	2	7Mbps	

Table 42

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	3.2	1514	pfifo	10	TCP		50.6
192.168.2.3	192.168.1.7	20	3.5	1514	RED		TCP		45.7

Table 43

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	3.2Mbps	20Mbytes				50.6
192.168.1.7	3.5Mbps	20Mbyte		10		45.7

Table 44

Table 43 and Table 44 show the results. Once again TCP traffic shows fairness. At the receivers we get 3.5Mbps and 3.2 Mbps for flow 1 and flow 2. These results are as expected since the TCP window size is controlling the sending rate of the sources and its below the available bandwidth.

Test 3

Still using the same scenario as above with the difference that the one source transmits a TCP and the other UDP traffic. Flow 1 transmits UDP traffic and flow 2 transmits TCP traffic.

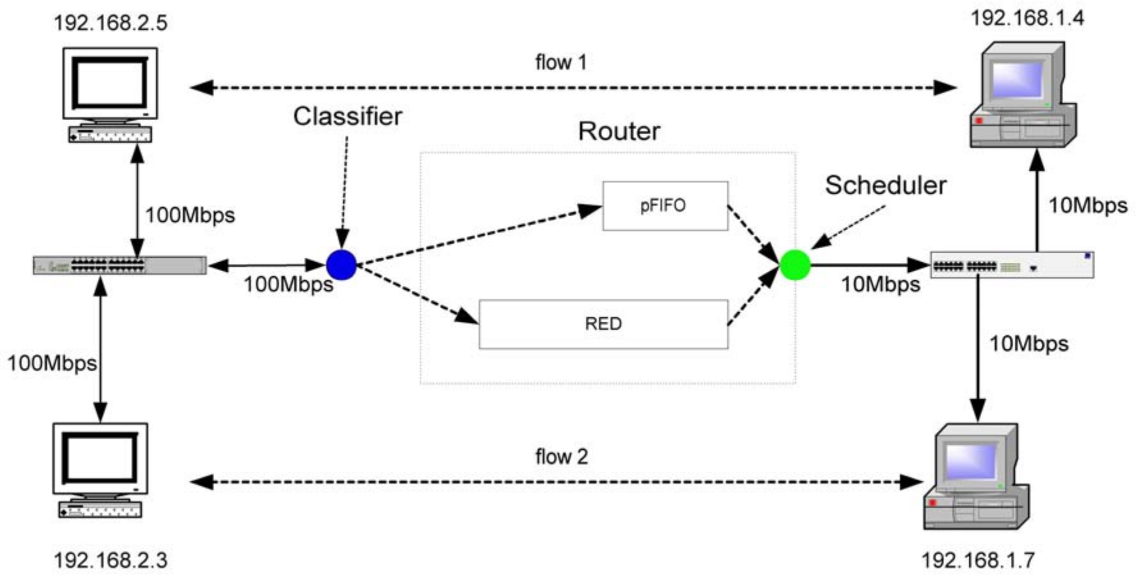


Figure 49: Block Diagram of EF PHB

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	300K	1	3Mbps	
192.168.2.3	U32	CBQ	5	700K	2	7Mbps	

Table 45

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	7	1514	pfifo	10	UDP	14267	22.9
192.168.2.3	192.168.1.7	20	4.1	1514	RED		TCP		39

Table 46

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	4.8 Mbps	13.7Mbytes	1.204	4518	14267	22.9
192.168.1.7	4.1Mbps	20Mbyte				39.2

Table 47

Table 46 and Table 47 show the results of the test. Once again, we can see that the UDP traffic steals traffic from the TCP traffic.

Test 4

In this test we repeat test 2 with focus on the queue behaviors. The setup parameters have been changed. We are allocating 5Mbps per flow, and we set the same class priorities. Both of the sources are transmitting TCP traffic. The TCP window size is 64K bytes at the sources and the receivers.

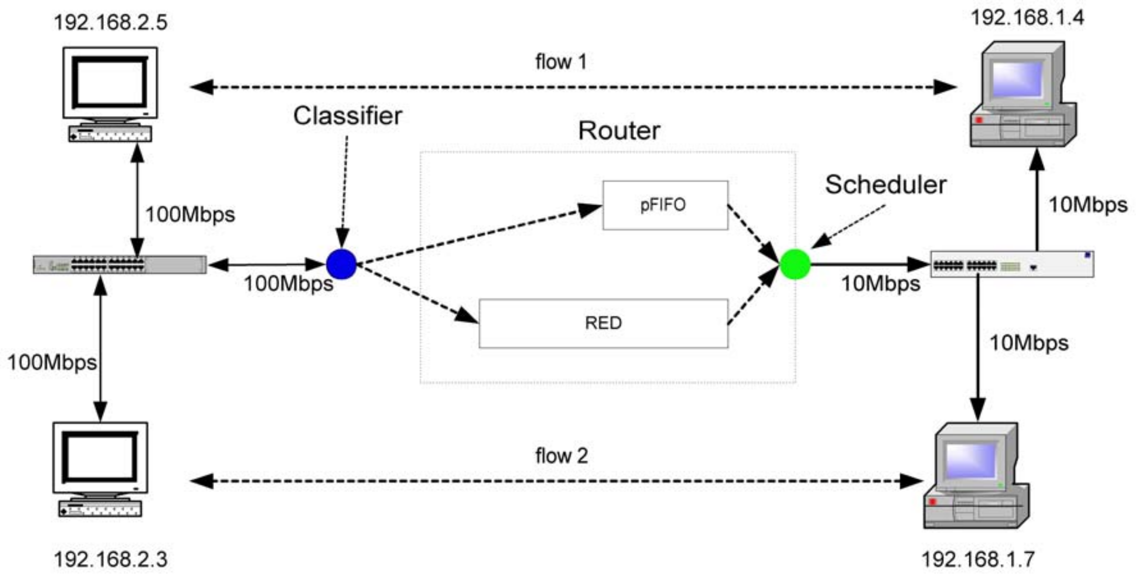


Figure 50: Block Diagram of EF PHB

The weights of the classes are the same 500K each class.

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	500K	1	5Mbps	
192.168.2.3	U32	CBQ	5	500K	2	5Mbps	

Table 48

The parameters for the RED queue are the following:

Limit 60KB

Maximum 45KB

Minimum 15KB

Probability 0.1

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets Transmitted	Time (s) Duration
192.168.2.5	192.168.1.4	20	3.6	1514	pfifo	10	TCP		44.2
192.168.2.3	192.168.1.7	20	3.4	1514	RED		TCP		47.0

Table 49

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	3.6Mbps	20Mbytes		0		46.5
192.168.1.7	3.4Mbps	20Mbyte		107		47.1

Table 50

In this test we observe some losses over flow 2, (107 lost packets). These packets are caused by RED since it is dropping packets based on the probability that we have assigned to the queue length. Figure 51and Figure 52 show the outstanding packets of both sources.

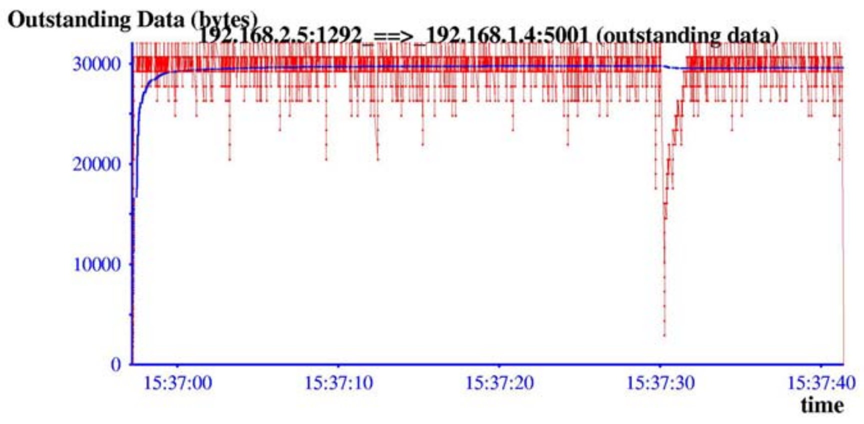


Figure 51: Outstanding Data

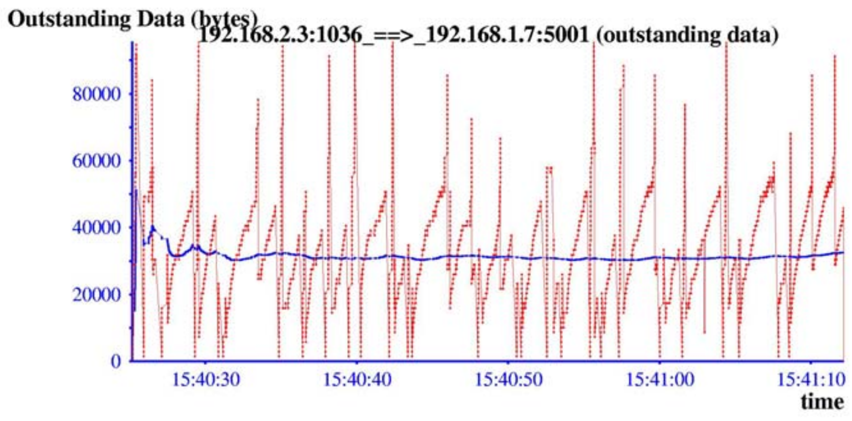


Figure 52: Outstanding Data

Figure 53 and Figure 54 show the Round Trip Time of both queues. We can see that the pFifo queue has larger RTT than the RED queue.

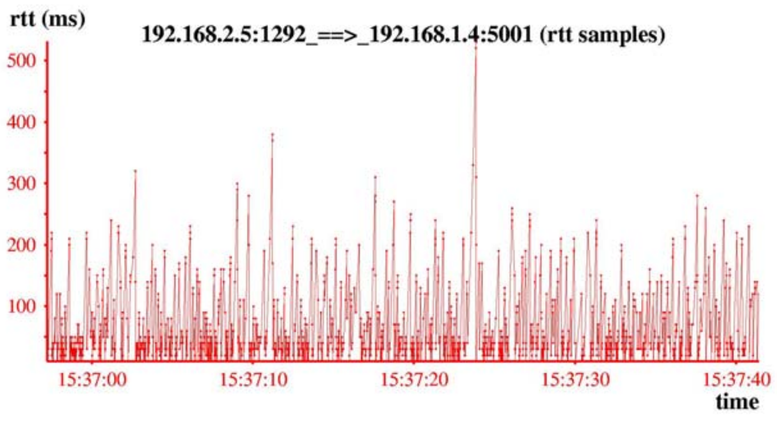


Figure 53: RTT of Flow 1

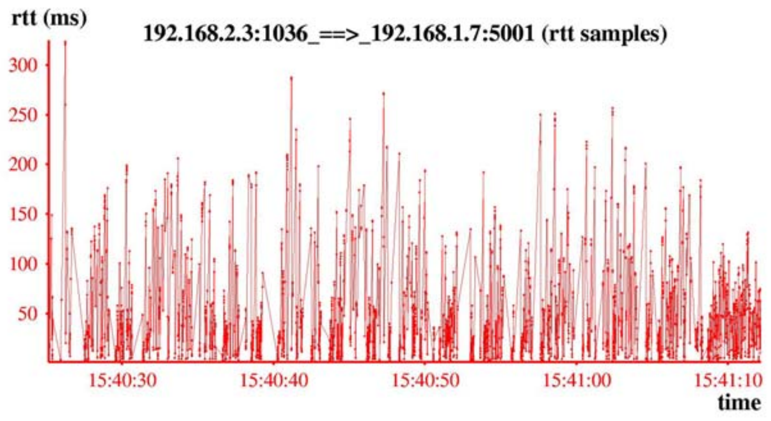


Figure 54: RTT of Flow 2

The TCP behavior on pFifo is straightforward. The TCP source sends packets based on the TCP window size and if the rate is higher than what the pFifo can sustain then the queue drops the packets. In this case we don't have packet drops in pFifo but we do have in RED. The drops in RED queue are expected, since after a certain threshold, RED has a certain probability that start dropping packets.

Note that the RTT time on both queues varies. On pfifo the RTT is larger than the RED. And that's again was expected. The RTT time is defined by how large the queue size is and since the pFifo is a fixed size then the RTT is fixed. On the other hand, the queue size of RED queue varies based on the mean queue size.

Test 5

In test 5 we used exactly the same parameters that we used in Test 4 with the exception that the TCP window size here is 128K bytes on both ends. We have increased the TCP window size in order to increase the throughput of the TCP traffic.

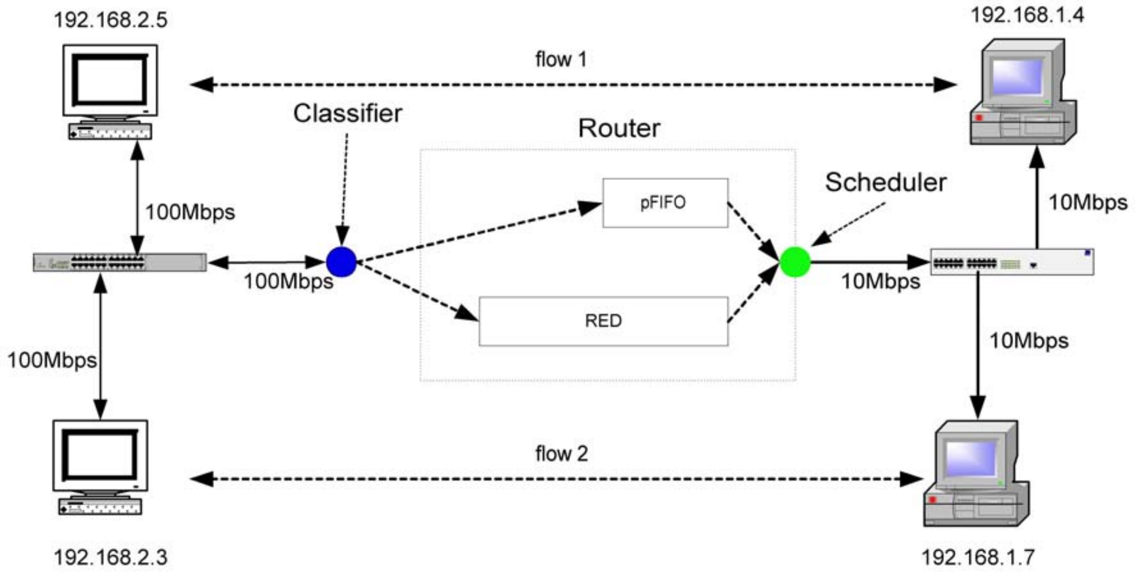


Figure 55: Block Diagram of EF PHB

Router Setup Parameters							
Source IP	Classifier	Scheduler	Class Priority	Class Weight	Filter Priority	Bandwidth	Notes
192.168.2.5	U32	CBQ	5	500K	1	5Mbps	
192.168.2.3	U32	CBQ	5	500K	2	5Mbps	

Table 51

Source Results									
Source IP	Destination IP	Transfer Mbytes	Rate Mbps	Frame Length	Queue Type	Queue size	Traffic Type	Packets lost	Time (s) Duration
192.168.2.5	192.168.1.4	20	3.6	1514	pfifo	50	TCP	11	44.6
192.168.2.3	192.168.1.7	20	3.4	1514	RED		TCP	21	46.7

Table 52

Destination Results						
Destination IP	Incoming Traffic	Data Transferred	Jitter Delay (ms)	Packet Loss	Total Packets	Time (s) Duration
192.168.1.4	3.6Mbps	20Mbytes		11		44.6
192.168.1.7	3.4Mbps	20Mbyte		21		46.7

Table 53

Table 52 and Table 53 show the results. With both sources transmitting TCP traffic we get 3.6Mbps and 3.4 Mbps for flow 1 and flow 2. Here we can see that packets were dropped at the pFIFO queue. This happens since we have increased the window size of the TCP.

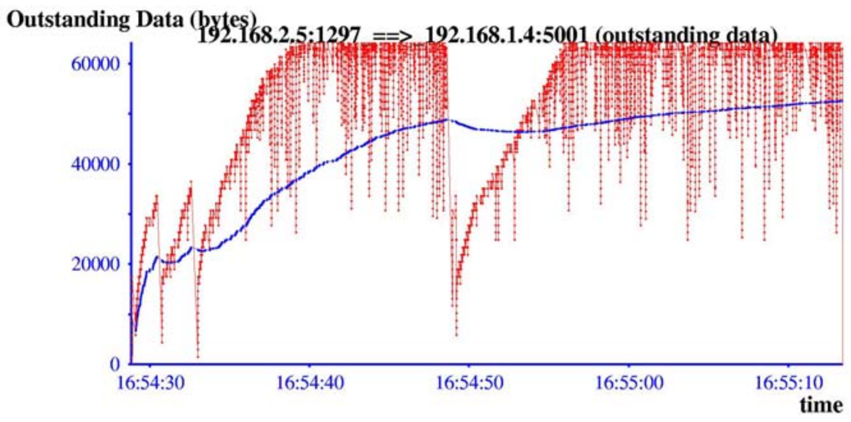


Figure 56: Outstanding Data of Flow 1

Figure 56 shows the slow start of the TCP and then shows the packets that are dropped. That's where the source starts sending at lower rates and start congestion avoidance algorithm.

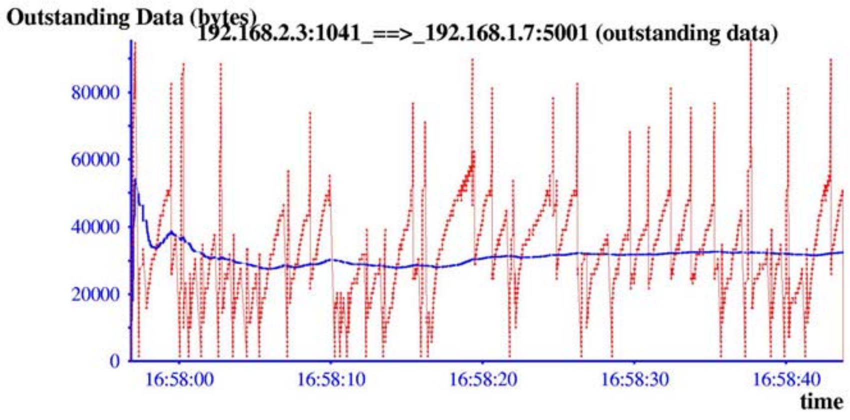


Figure 57: Outstanding Data of Flow 2

Figure 57, on the other hand shows a different behavior. There are dropped random drops based on the probability drop of the RED queue. This shows some smoothness over the queue. We don't get big variations in the queues as in the pFifo.

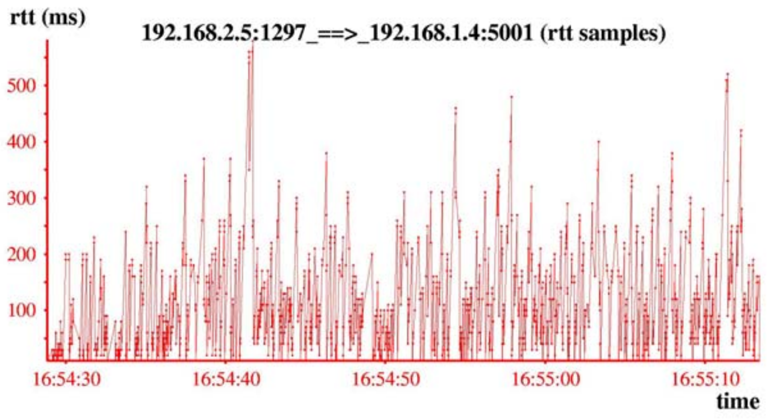


Figure 58: RTT of Flow 1

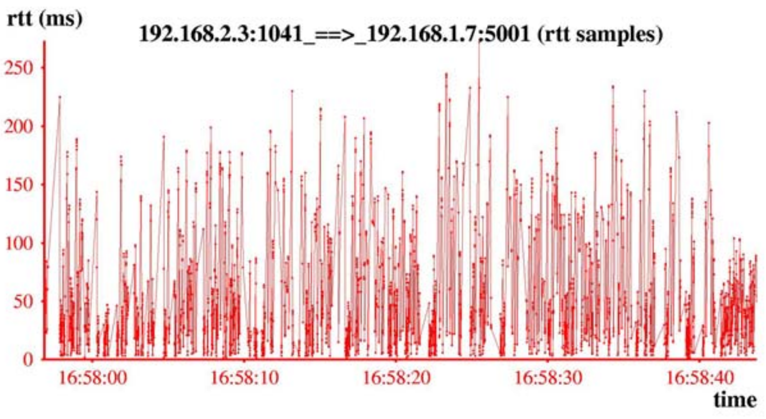


Figure 59: RTT of Flow 2

Figure 58 and Figure 59 show the RTT of the pFifo and RED queues respectively. It's obvious the RED queue get smaller RTT because the mean queue size of the queue is smaller. The maximum RTT of the RED is 250ms and the pFifo is 500ms.

5.3 Summary of Results

In this section we present a summary of the results. At the right most column you can find some comments for each test.

Results of Scenario 1 (scheduler CBQ)							
	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type	Results	Comments
Test 1							
	5	500K	1	pFifo	UDP	4.5Mbps	As Expected
	5	500K	1	pFifo	UDP	4.6Mbps	As Expected
Test 2							
			100			4.5Mbps	No effect as expected
			1			4.6Mbps	No effect as expected
Test 3							
	1		1			7Mbps	As Expected
	5		1			2.1Mbps	As Expected
Test 4							
	4		1			7Mbps	Not sensitive to priority level. Rather to higher prio.
	5		1			2.2Mbps	
Test 5							
	5	800K	1			6.6Mbps	Close Enough
	5	200K	1			2.5Mbps	Close enough
Test 6							
	5	500K	1		UDP	6.9Mbps	Expected
	5	500K	1		TCP	3.8Mbps	Expected
Test 7							
	5	500K	1		TCP	3.5Mbps	Expected
	5	500K	1		TCP	3.4Mbps	Expected

Table 54

Results of Scenario 2								
	Scheduler	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type	Results	Comments
Test 1								
	CBQ	5	200K	2	TBF	UDP	1.1Mbps	As Expected
		5	800K	1	pFifo	UDP	7.0Mbps	As Expected
Test 2								
						UDP	1Mbps	As Expected
						TCP	5.7Mbps	As Expected

Table 55

Results of Scenario 3								
	Scheduler	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type	Results	Comments
Test 1								
	PQ	PQ1	200K	5	pFifo	UDP	7Mbps	As Expected
		PQ2	800K	5	pFifo	UDP	2.2Mbps	As Expected
Test 2								
	PQ	PQ2	200K	5	pFifo	UDP	2.2Mbps	As Expected
		PQ1	800K	5	pFifo	UDP	7Mbps	As Expected

Table 56

Results of Scenario 4								
	Scheduler	Class Priority	Class Weight	Filter Priority	Queue Types	Traffic Type	Results	Comments
Test 1								
	CBQ	5	300K	2	pFifo	UDP	2.7Mbps	As Expected
		5	700K	1	RED	UDP	6.4Mbps	As Expected
Test 2								
						TCP	3.2Mbps	As Expected
						TCP	3.5Mbps	As Expected
Test 3								
		5		1		UDP	4.8Mbps	As Expected
		5		1		TCP	4.1Mbps	As Expected
Test 4(TCP Window size 64K)								
		5	500K	1		TCP	3.6Mbps	As Expected
		5	500K	1		TCP	3.4Mbps	As Expected
Test 5 (TCP Window size 128K)								
		5	500K	1		TCP	3.6Mbps	As Expected
		5	500K	1		TCP	3.4Mbps	As Expected

Table 57

5.4 Recommendations for selecting between various network functions and settings

This section provides some practical recommendation regarding the selection between various networking functions and their settings, based on different user demands. . Analyzing the results we have obtained from the experiments we can see that certain queuing disciplines can favor certain types of traffic.

We will first provide two scenarios to show that for different user/application demands there may be a need for differentiation of services and that there are various ways to implement differentiation of services. We will then generalize some recommendations for various networking conditions.

Scenario 1

A company has a network link of capacity 10Mbps and the IT manager wants to limit the employees for using real time applications such as video on demand, videoconferencing and etc to 2Mbps. He doesn't want his employees to use most of the company's bandwidth and not having enough bandwidth for the company's Web Server.

Under normal TCP/IP, we observe, as expected[1], that UDP traffic is out beating TCP traffic in most of the scenarios; see Chapter 5. We can provide some fairness in various ways.

Recommendation 1:

One solution for this scenario is to use DiffServ and create two classes by using CBQ discipline. Class A will be for real applications (UDP traffic) and will be allocated 2Mbps bandwidth. A pFifo queuing discipline can be used for Class A. In Class B can be assigned 8Mbps for all the other applications. In Class B a RED queuing discipline can be used. This solution eliminates the problem of starvation at the peak times, but in case Class B doesn't require the 8Mbps that has been assigned, then Class A can utilized the excess (above 2 Mbits/sec) bandwidth.

Recommendation 2

Another approach of this scenario is to use Token Bucket filter, TBF, for Class A. By using TBF we restrict Class A, real time applications, to use any bandwidth from Class B. TBF acts as a limiter and doesn't allow to the UDP traffic to use more traffic than it supposed to.

[1] Because TCP is controlled by the state of the network, it backs off when congestion is sensed. On the contrary UDP sends packets into the network as the application demands.

Scenario 2

A company has a network link of capacity 10Mbps and the IT manager wants to differentiate the services of the company based on the departments. He believes that the marketing department should have more priority on the Internet than the other departments.

Recommendation 1

He can implement this scenario by using Priority Queue. He has to assign the Marketing Department on the first queue and the rest to the second queue with lower priority. The packets on the second queue will be serviced only when the first queue is empty. This way the Marketing Department gets a higher priority over the others.

Recommendation 2

Another way to implement this scenario is to use CBQ, Class Based Queuing, and assign different priorities for each class. We need to create at least two classes and assign one for the Marketing Department and one for the other departments. We could have created more classes and more priorities if we wanted to. CBQ allows to the IT manager to isolate or create more priorities within the company. Even more he could use different queuing disciplines such as pFIFO for the marketing department and RED for the others.

Note: As can be seen, there are various queuing disciplines that can implement differentiation of service. The following are some general recommendations, derived from the experimentation in this thesis:

General recommendations

- PFIFO is best used for UDP traffic. The queue length should be small in order to obtain smaller RTT.

- RED is best used for TCP traffic. As we know RED drops packets based on a probability we assign. This is good for TCP traffic to avoid sharp fluctuations and avoid congestion.

- TBF is best used where we want to limit UDP traffic.

6 Conclusions

In this thesis, we implemented a differentiated services pilot network in Linux environment and investigated the performance of network functions that provided quality of service.

We observed that the basic functions required for differentiated service could be implemented under the proposed Linux operating system. The observed results show that a number of different approaches maybe taken to implement differentiation of service.

We cannot generalize the results since the available topology is limited, however some observations and recommendations have been offered. It's strongly recommended to expand the current topology to a more realistic one, including LAN and WAN segments and generalize the observations and recommendations.

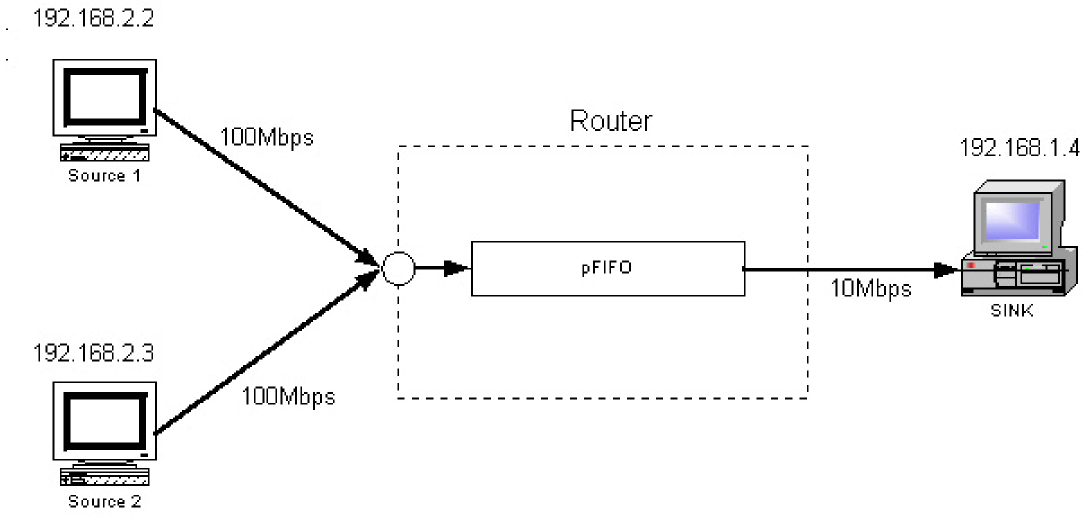
Basically, our investigation show that differentiation of services can be achieved by using variety of network functions and parameters, and for a simple network topology some conclusions and recommendations are drawn. However, large-scale applications are a much more complex task requiring further study.

In the future we also want to investigate traffic management. In particular we expect to implement congestion control strategies that are more suitable for differentiated services such as the Integrated Switching strategy [32].

7 Appendix I

Scenarios

The following scenarios aim at verifying the expected behavior of the TCP/IP implementation and the traffic Control (TC) under this Linux implementation. In general, expected behavior has been validated.



Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				pfifo	50	TCP	60
192.168.2.5	192.168.1.4				pfifo	50	TCP	60
192.168.2.3	192.168.1.4				pfifo	100	TCP	60
192.168.2.5	192.168.1.4				pfifo	100	TCP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	3.6Mbps	3.6Mbps	119	
192.168.2.5	3.5Mbps	3.5Mbps		
192.168.2.3	3.6Mbps	3.6Mbps	12	
192.168.2.5	3.6Mbps	3.6Mbps		

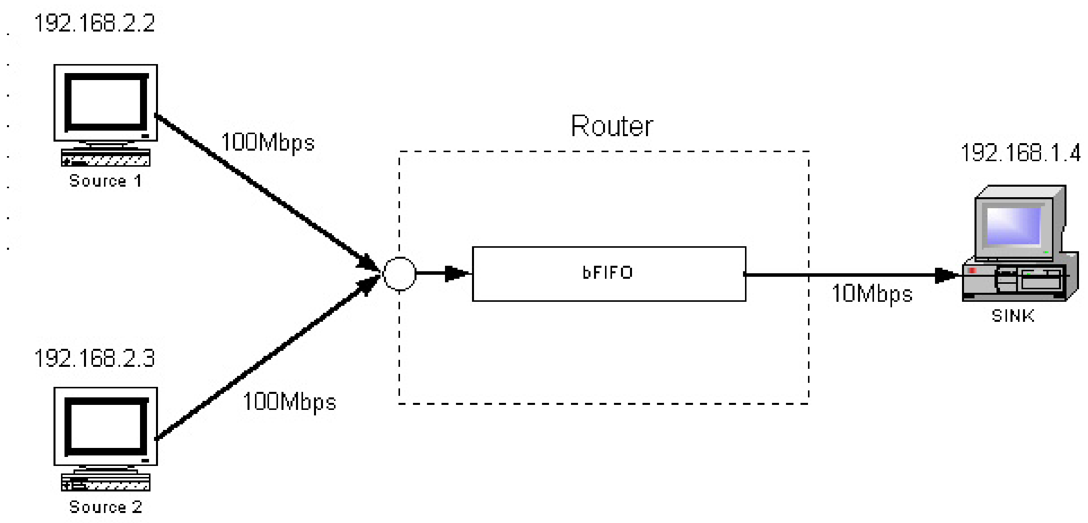
Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				pfifo	50	TCP	60
192.168.2.5	192.168.1.4				pfifo	50	UDP	60
192.168.2.3	192.168.1.4				pfifo	100	TCP	60
192.168.2.5	192.168.1.4				pfifo	100	UDP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.4	65Kbps	64.9Kbps	71	
192.168.2.5	10Mbps	9.1Mbps	4826	
192.168.2.4	71.5Kbps	71.4Kbps	48	
192.168.2.5	10Mbps	9.1Mbps	4727	

Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				pfifo	50	UDP	60
192.168.2.5	192.168.1.4				pfifo	50	UDP	60
192.168.2.3	192.168.1.4				pfifo	100	UDP	60
192.168.2.5	192.168.1.4				pfifo	100	UDP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	9.5Mbps	2.5Mbps	37231	
192.168.2.5	10Mbps	6.6Mbps	18133	
192.168.2.3	9.4Mbps	2.6Mbps	36276	
192.168.2.5	10Mbps	6.6Mbps	17915	

Test 2 (bfifo)



Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				bfifo	75K	TCP	60
192.168.2.5	192.168.1.4				bfifo	75K	TCP	60
192.168.2.3	192.168.1.4				bfifo	150K	TCP	60
192.168.2.5	192.168.1.4				bfifo	150K	TCP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	3.4Mbps	3.4Mbps	105	
192.168.2.5	3.1Mbps	3.1Mbps		
192.168.2.3	3.3Mbps	3.3Mbps	14	
192.168.2.5	3.7Mbps	3.7Mbps		

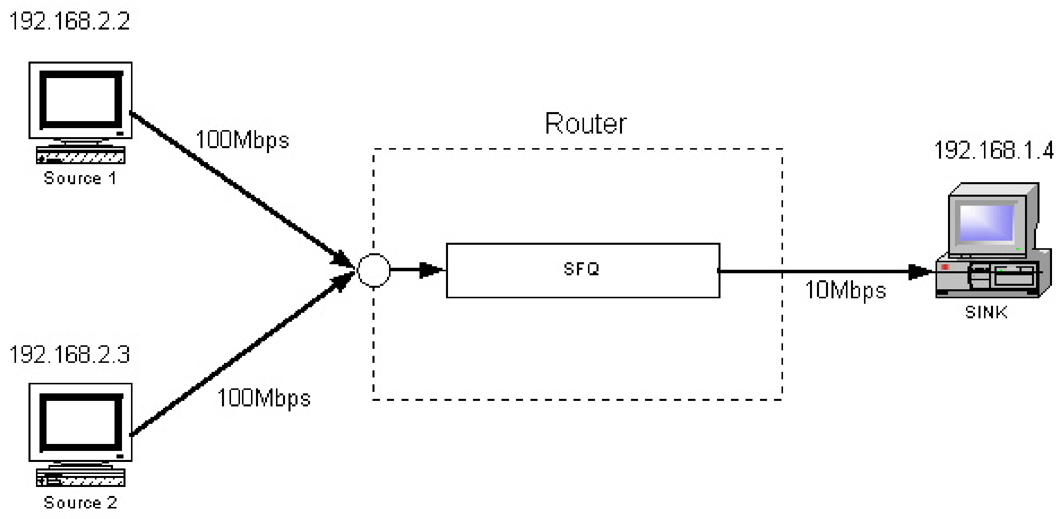
Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				bfifo	75K	TCP	60
192.168.2.5	192.168.1.4				bfifo	75K	UDP	60
192.168.2.3	192.168.1.4				bfifo	150K	TCP	60
192.168.2.5	192.168.1.4				bfifo	150K	UDP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	93Kbps	92.9Kbps	71	
192.168.2.5	10Mbps	9.1Mbps	4832	
192.168.2.3	89.1Kbps	89.1Kbps	73	
192.168.2.5	10Mbps	9.1Mbps	4790	

Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				bfifo	75K	UDP	60
192.168.2.5	192.168.1.4				bfifo	75K	UDP	60
192.168.2.3	192.168.1.4				bfifo	150K	UDP	60
192.168.2.5	192.168.1.4				bfifo	150K	UDP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	9.5Mbps	2.6Mbps	36881	
192.168.2.5	10Mbps	6.6Mbps	18160	
192.168.2.3	9.5Mbps	2.6Mbps	36539	
192.168.2.5	10Mbps	6.6Mbps	18136	

Test 3 (SFQ)



Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				SFQ		TCP	60
192.168.2.5	192.168.1.4				SFQ		TCP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	3.5Mbps	3.5Mbps	0	
192.168.2.5	3.5Mbps	3.5Mbps	0	

Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				SFQ		TCP	60
192.168.2.5	192.168.1.4				SFQ		UDP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	851Kbps	850Kbps	0	
192.168.2.5	10Mbps	8.3Mbps	9162	

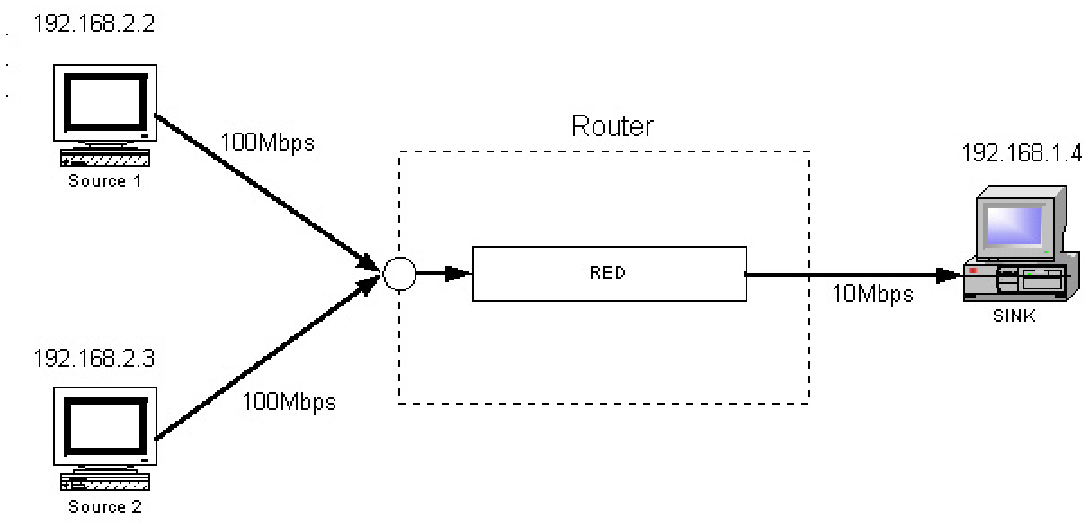
*381 datagrams received out of order

Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				SFQ		UDP	60
192.168.2.5	192.168.1.4				SFQ		UDP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	9.5Mbps	3.6Mbps	31165	
192.168.2.5	10Mbps	5.5Mbps	23858	

84 out of order

Test 4 (RED)



Limit 50000 bytes

Min 15000 bytes

Max 45000 bytes

Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				RED		TCP	60
192.168.2.5	192.168.1.4				RED		TCP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	6.1Mbps	6.1Mbps	51	
192.168.2.5	350Kbps	350Kbps	51	

Parameters								
Source IP	**Dest IP**	**Load %**	**Load Mbps**	**Frame Length**	**Queue**	**Queue size**	**Traffic type**	**Duration (seconds)**
192.168.2.3	192.168.1.4				RED		TCP	60
192.168.2.5	192.168.1.4				RED		UDP	60

Results				
Source IP	**Incoming Traffic**	**Outgoing traffic**	**Packet Loss**	**Average Delay**
192.168.2.3	78.5Kbps	78.5Kbps	109905	
192.168.2.5	10Mbps	4.6Mbps	28952	

8 Acronyms

ADU Application Data Unit

AF Assured Forwarding

AS Autonomous System

ATM Asynchronous Transfer Mode

BGP Border Gateway Protocol

CBS Committed Burst Size

CE Congestion Experienced

CIR Committed Information Rate

CL Controlled Load

COPS Common Open Policy Service

CRC Cyclical Redundancy Check

CSMA/CD Carrier Sense Multiple Access with Collision Detection

cwnd Congestion window, one of the state variables maintained by TCP

CWR Congestion window reduced

Deficit-WRR Deficit Weighted Round Robin

DNS Domain Name System

DSAP Destination Service Access Point

DSCP Differentiated Services CodePoint, a six-bit field in the IP header used to en-code the per-hop-behavior

EBS Excess Burst Size

EF Expedited Forwarding, one of the Differentiated Services defined by IETF [JNP99]

E-LSP EXP-Inferred Packed Scheduling Class Label Switched Path

ER Edge Router

FF Fixed Filter, one of the sender filters supported by RSVP

FIFO First In First Out

FTP File Transfer Protocol

GPS Generalized Processor Sharing, an ideal scheduler

GS Guaranteed Service, one of the Integrated Services defined by IETF

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force, see http://www.ietf.org

IHL IP Header Length, one of the fields of the IP header

IP Internet Protocol

ISDN Integrated Services Digital Network

IS-IS Intermediate System - Intermediate System

ISP Internet Service Provider

LAN Local Area Network

LDAP Lightweight Directory Access Protocol

L-LSP Label-Only-Inferred Packet Scheduling Class Label Switched Path

LSA Link State Attribute, a type of control packet distributed by OSPF

LSP Label Switched Path

MPLS Multiprotocol Label Switching

MSS Maximum Segment Size

MTU Maximum Transfer Unit

NFS Network File System

OSPF Open Shortest Path First

PDB Per Domain Behaviour

PHB Per Hop Behaviour

PDP Policy Decision Point

PDU Protocol Data Unit

PEP Policy Enforcement Point

PGPS Packetized Generalized Processor Sharing

PIR Peak Information Rate

PNNI Private Network-Network Interface, the routing protocol used in ATM Net-works

PPTP Point-to-Point Tunneling Protocol

PQ Priority Queuing, a priority-based scheduler

PS Processor Sharing, an ideal scheduler

QoS Quality of Service

RED Random Early Detection, a buffer acceptance algorithm

RIP Route Information Protocol

RPC Remote Procedure Call

RSVP Resource Reservation Protocol

RTCP RTP Control Protocol

RTP Real Time Transport Protocol

RTSP Real Time Streaming Protocol

RTT round trip time

rwin receiving window, a state variable maintained by TCP

SCFQ Self Clocked Fair Queueing

SDES Source Description, used by RTCP

SE Shared Explicit filter, one of the sender filters supported by RSVP

SIP Session Initiation Protocol

SNMP Simple Network Management Protocol

SSRC Synchronization source identifier, used by RTP

sstresh slow start threshold, state variable maintained by TCP

swin	sending window, state variable maintained by TCP
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
THL	TCP Header Length, part of the TCP header
ToS	Type of Service
TTL	Time To Live
UDP	User Datagram Protocol
WAN	Wide Area Network
WF	Wild-card filter, one of the sender filters supported by RSVP
WFQ	Weighted Fair Queueing
WRED	Weighted Random Early Detection, a buffer acceptance algorithm supporting several packet dropping preferences
WRR	Weighted Round Robin, a scheduler suitable for fixed-length packets

9 Bibliography

[1] M. Luoma, QoS and queuing disciplines, traffic and admission control, submitted for publication.

[2] *F. Baumgartner, T. Braun, P. Habegger,* University of Berne Proc. of the 8th IFIP Conference on High Performance Networking - HPN'98, September 1998.

[3] K. Kilkki. *Differentiated services for the Interne*t. MacMillan Technology Series, 1999.

[4] M. Shreedhar and G. Vargese. Efficient fair queuing using deficit round robin. In *Proc. ACM SICOGMM'9*5, pages 231–242, 1995.

[5] W. Almesberger, Linux Network Traffic Control- Implementation Overview, Technical Report EPFL ICA, April 1999

[6] . Braun, H. Einsiedler, M. Scheidegger, K. Jonas, H. *Sttgen: A Linux Implementation of a Differentiated Services Router*, submitted for publication

[7] W. Almesberger, J. Hadi Salim, and A. Kuznetsov. Differentiated services on Linux. Internet draft, draft-almesberger-wajhak-diffserv-linux-00.txt, work in progress, February 1999.

[8] W. Stevens. *TCP/IP Illustrated, volume 1 : The protocol*s. Addison-Wesley, 1994.

[9] P. Britaain, A. Farrel, "MPLS TRAFFIC ENGINEERING: A CHOICE OF SIGNALING PROTOCOLS" Data Connection, January 17,2000.

[10] Quality of Services Networking -Chapter 46 of Internetworking Technology Overview June 1999. Accessible at http://www.cisco.com

[11] Feng, D. Kandlur, D. Saha, K. Shin, "BLUE: A New Class of Active Management Algorithms" Department of EECS, University of Michingan and IBM T.J Watson Research Center.

[12] C. Metz, "IP QOS: Traveling in First Class on the Internet," IEEE Internet Computing April 1999, Accessible at http://computer.org/internet

[13] Nortel/Bay Networks, *IP* ***QoS*** *--A Bold New Network* , white paper, Accessible at http://www.nortelnetworks.com

[14] Congestion Avoidance Overview. Accessible at http://www.cisco.com

[15] U. Schafer, "Investigation of the Differentiated Services Concept in the Future Internet." University of Wurzburg.

[16] IP Traffic Engineering for Carrier Networks: Using Constraint-Based Routing to Deliver New Services. Accessible at http://www.nortelnetworks.com

[17] QoS Protocols and Architectures. Accessible at http://www.qosforum.com

[18] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "*A model based TCP-friendly rate control protocol*," in Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, June 1999. 20

[19] S. Radhakrishman, Linux- Advanced Networking Overview, V1, Department of Electrical Engineering & Computer Science, University of Kansas

[20] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weis-s. An architecture for differentiated services. Internet RFC 2475, December 1998.

[21] S. Floyd and V. Jacobson. Random early detection for congestion avoidance. *IEEE/ACM Transactions on Networkin*g, 1(4):397–413, August 1993.

[22] S. Floyd. RED (Random Early Detection) queue management. available from http://www-nrg.ee.lbl.gov/floyd/red.html, 1999.

[23] R. Guerin, S. Kamat, V. Peris, and R. Rajan. Scalable QoS provi-sion through buffer management. In *SIGCOMM 9*8, pages 29–40, Vancouver, Canada, August 1998.

[24] R. Guerin and V. Peris. Quality of service in packet networks: basic mechanisms and directions. *Computer Network*s, 31:169–189,1999.

[25] V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding PHB. Internet RFC2598, June 1999.

[26] E. Ellesson and S. Blake. A proposal for the format and se-mantics of the TOS byte and traffic class byte in IPv4 and IPv6 headers. Internet Draft, draft-ellesson-tos-00.txt, November 1997. (Work in progress). September 1997.

[27] Dovrolis, C., Stiliadis, D., and Ramanathan, P. Proportional Differentiated Services. In Proceedings of SIGCOM (October 1999), vol. 29

[28] RFC2475; Blake, Steven; Black, David; Carlson, Mark; Davies, Elwyn; Wang, Zheng; Weiss, Walter. An Architecture for Differentiated Services, IETF, December 1998.

[29] Feng, W., Kandlur, K., Saha, D., and Shin, K. Understanding tcp dynamics in an integrated services Internet. In NOSSDAV '97 (MAY 97)

[30] Feng, W., Kandlur, K., Saha, D., and Shin, K. Adaptive packet marking for providing differentiated services on the Internet. In proceedings of 1998 International conference on Network Protocols (INCP '98) (October 1998)

[31] A. Pitsillides, P. Ioanou, L. Rossides, Congestion Control using Non-linear Control Theory: Integrated dynamic Congestion Controller, ISCC 2001, Hammanet, Tunisia, July 1-3, 2001

[32] L. Rossides, S Kohler, A. Pitsillides, T-G. Phuoc, Fuzzy RED: Congestion control for TCP/IP Diff-Serv, Melecon 2000, Limasol, May 29-31, 2000