University of L'Aquila

Erasmus Mundus Mathmods Program

Master Thesis

Modeling of traffic congestion in a QoS Based service provider network

Student:
Amirthalakshmi Veeraraghavan

Advisor:
Prof. Alessandro D'Innocenzo
Ing.Ionta Tiziano

A thesis submitted in fulfilment of the requirements for the degree of M.Sc in Mathematical modeling in Engineering: Theory, Numerics and Application

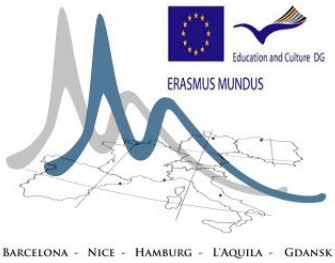
in the

Department of Information Engineering, Computer Science and Mathematics

With the support from the Erasmus Mundus Program of the European Union and Telecom Italia

Data discussione tesi: 11.09.2014

Università degli Studi di L'Aquila



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Declaration of Authorship

I, *Amirthalakshmi Veeraraghavan*, declare that this thesis titled, '*Modeling of traffic congestion in a QoS Based service provider network*' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITY OF L'AQUILA

Abstract

Prof.Alessandro D'Innocenzo
Department of Information Engineering, Computer Science and Mathematics

M.Sc,Mathemtatical Modeling in Engineering

Modeling of traffic congestion in a QoS Based service provider network

by Amirthalakshmi Veeraraghavan

The main objective of this Master thesis is to create a mathematical and simulative environment to formally model the effect of QoS on the dynamics of TCP packet flows belonging to different end-user services (i.e. http, ftp, mailing and video streaming) transiting through a Service Provider backbone network. Our model takes into account network topology, routing (a simplified version of OSPFv2 on Ipv4), TCP (a full model of the TCP-Sack protocol),QoS for packets priority and different end-user services.The aim of this research is to provide a framework to analyze quantitatively the effect of QoS implemented in Telecom Italia on packet loss and throughput of different services and compare with different QoS protocols.In particular,we study the effect of packet loss and throughput on QoE (Quality of Experience) for video streaming in the presence of different QoS Networking methods.

Acknowledgements

Firstly, I would like to present my sincere gratitude to my supervisor Prof.Alessandro D'Innocenzo who guided, supported me in each and every phase of Master thesis and gave an opportunity to work with this project. It was an extreme honour and pleasure that I had a chance to work with him.

I wish to express my thanks to my co-supervisors Ing.Ionta Tiziano from Telecom Italia and Ing.Alessandra Falasca from CiSCO, for being so kind,helpful and taking their time in clarifying all my doubts.

Specially, I want to thank Professor Rubino who made all this possible.

My gratitude will also go to my lovely friends, MathMods colleagues for creating such a stimulating atmosphere. It would have been impossible to finish my master thesis without their help.

Finally, I am greatly thankful to my parents,uncle and my brother who are also my best friends. Thanks for always supporting and motivating me.

Contents

List of Figures

Abbreviations

RTT	**R**ound **T**rip **T**ime
DDD	**D**rop **D**etection **D**elay
TCP	**T**ransfer **C**ontrol **P**rotocol
QoS	**Q**uality **o**f **S**ervice
QoE	**Q**uality **o**f **E**xperience
SACK	**S**elective **ACK**knowledgement
LAN	**L**ocal **A**rea **N**etwork
WAN	**W**ide **A**rea **N**etwork
IP	**I**nternet **P**rotocol
ACL	**A**ccess **C**ontrol **L**ist
PQ	**P**rioity **Q**ueuing
CQ	**C**ustom **Q**ueuing
CBWFQ	**C**lass **B**Based **W**eighted **F**air **Q**ueuing
CAR	**C**ommitted **A**ccess **R**ate
NBAR	**N**etwork **B**ased **A**pplication **R**ecognition
URL	**U**niform **R**esource **L**ocator
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
PBR	**P**olicy **B**ased **R**outing
ToS	**T**ype **o**f **S**ervice
FIFO	**F**irst **I**n **F**irst **O**ut
RED	**R**andom **E**arly **D**etection
WRED	**W**eighted **R**andom **E**arly **D**etection
GTS	**G**eneric **T**raffic **S**haping
LFI	**L**ink **F**ragmentation and **I**nterleaving
RTP	**R**eal-time **T**ransport **P**rotocol

MIME	**M**ulti-Purpose **I**nternet **M**ail **E**xtensions
MDRR	**M**odified **D**eficit **R**ound **R**obin
PSNR	**P**eak **S**ignal to **N**oise **R**atio
PVQ	**P**erceived **V**ideo **Q**uality
VBR	**V**ideo **B**it **R**ate
PLR	**P**acket **L**oss **R**ate
MOS	**M**ean **O**pinion **S**core
APM	**A**pplication **P**erformance **M**etrics

Symbols

r_{h_f}	sending rate of high/Gold priority flows
r_{m_f}	sending rate of medium/Premium priority flows
r_{n_f}	sending rate of normal/Default priority flows
r_{lw_f}	sending rate of low priority flows
s_{h_f}	arrival rate of high/Gold priority flows
s_{m_f}	arrival rate of medium/Premium priority flows
s_{n_f}	arrival rate of normal/Default priority flows
s_{lw_f}	arrival rate of low priority flows
d_{h_f}	drop rate of high/Gold priority flows
d_{m_f}	drop rate of medium/Premium priority flows
d_{n_f}	drop rate of normal/Default priority flows
d_{lw_f}	drop rate of low priority flows
z_h	packet losses in high/Gold priority flows
z_m	packet losses in medium/Premium priority flows
z_n	packet losses in normal/Default priority flows
z_{lw}	packet losses in low priority flows
T^l	Transport delay
q_{h_f}	Number of bytes of flow f in high/Gold priority Queue
q_{m_f}	Number of bytes of flow f in medium/Premium priority flows
q_{n_f}	Number of bytes of flow f in normal/Default priority flows
q_{lw_f}	Number of bytes of flow f in low priority flows
B^l	Bandwidth of link l
w_f	size of congestion window
w^{adv}	advertised window size
$ssthr_f$	threshold size of congestion window for flow f

Chapter 1

Introduction

Data communication networks are highly complex systems, thus modeling and analyzing their behavior is quite challenging. The problem aggravates as networks become larger and more complex. The most accurate network models are packet-level models [5] that keep track of individual packets as they travel across the network. These are used in network simulators such as ns-2. Packet-level models have two main drawbacks: the large computational requirements (both in processing and storage) for large-scale simulations and the difficulty in understanding how network parameters affect the overall system performance. Aggregate fluid-like models [10],[9] overcome these problems by simply keeping track of the average quantities that are relevant for network design and provisioning (such as queue sizes, transmission rates, drop rates, etc).The main limitation of these aggregate models is that they mostly capture steady state behavior because the averaging is typically done over large time scales. For instance, detailed transient behavior during congestion control cannot be captured. Consequently, these models are unsuitable for a number of scenarios, including capturing the dynamics of short-lived flows.

Our approach to modeling computer networks and its protocols is to use hybrid systems which combine both continuous time dynamics and discrete-time logic. These models permit complexity reduction through continuous approximation of variables like queue and congestion window size, without compromising the expressiveness of logic-based models. The hybridness of the model comes from the fact that, by using averaging, many variables that are essentially discrete (such as queue and window sizes) are allowed to take continuous values. However, because averaging occurs over short time

FIGURE 1.1: No QoS

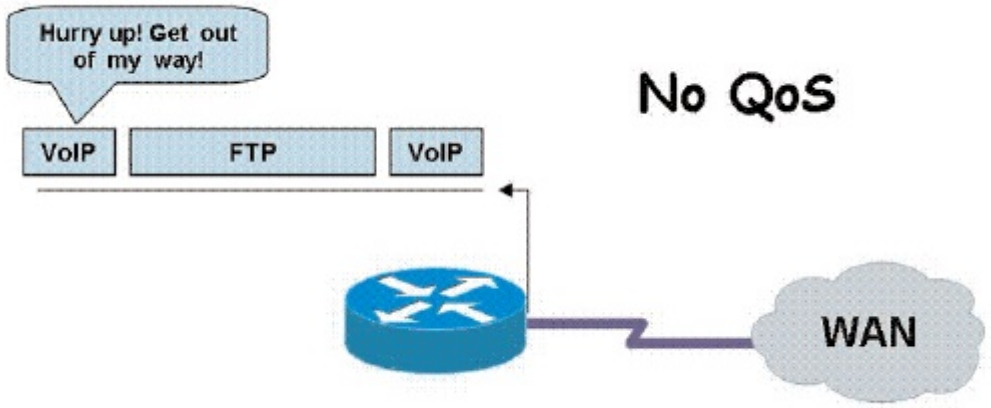
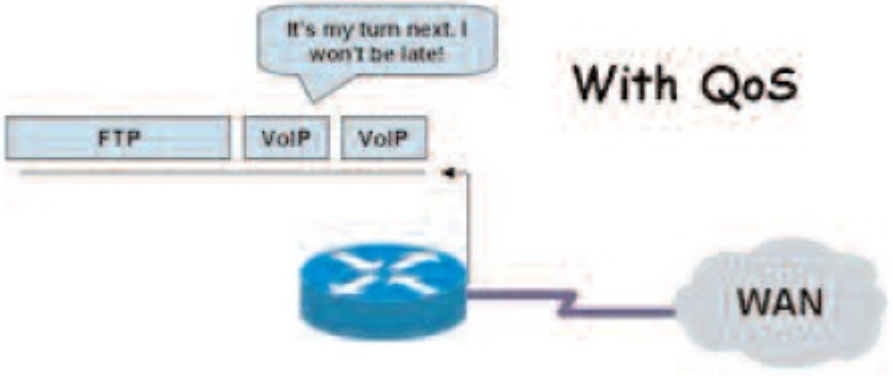


FIGURE 1.2: With QoS



intervals, one still models discrete events such as the occurrence of a drop and the consequent reaction (e.g.,congestion control).

Here, we have extended the framework in [3],[4] to take into account QoS Networking. QoS is the ability to treat packets differently as they transit a network device, based on the packet contents. The motivation for QoS comes from the user's complaints such as slower applications, jerky video and the out of sync audio and bad quality voice calls. Without QoS policies each service is given equal access to resources. In order for a company to efficiently utilize its network resources, it must identify which network traffic is critical and allocate appropriate resources to support those traffic streams. If voice is present in the network,it must get priority over all data streams; otherwise, the result could be intermittent voice quality complaints.Voice and video applications are delay- and jitter-sensitive.

The figures 1.1 and 1.2 show the difference between with QoS and No QoS. Without

QoS, a customer may experience delays in reaching the online catalog because of congestion caused by the video download. With QoS configuration, priority could be given to http packets that are sent to and from the corporate web server, while other web packets could be given lower priority. This would maintain a high level of resources for customer access while still providing some bandwidth to staff web traffic.With the mix of traffic present in today's networks, QoS ensures that the right applications get access to network resources first.

QoS can also be used to differentiate data packets from different application stream and provide access to resources according to policy.

As explained above, QoS is network centric performance indicator, while QoE (Quality of Experience) is a user centric performance indicator of the network. It is the most important factor for customers while choosing a network provider.To ensure superior QoE, it is important to understand the relationship between QoE and QoS.

In this project,a hybrid model of QoS implemented in Telecom Italia and Priority Queuing have been developed and metrics such as packet loss,delay and throughput were compared between the two. The models were also tested with the data from Telecom Italia. In addition to this, different ways to calculate QoE for video streaming services have been studied and implemented.

The report is organised as follows : Chapter 2 covers QoS concepts and methods; Chapter 3 covers the Queue dynamics, Hybrid Model of TCP-SACK,Hybrid model of Priority Queuing (a particular QoS Networking method) and MDRR(QoS implemented in Telecom Italia); Chapter 4 covers the MATLAB Simulink model of the QoS techniques mentioned before and their Simulation Results and Chaper 5 with concluding remarks and future work.

Chapter 2

Quality of Service Networking and Quality of Experience

2.1 Introduction to QoS

Quality of Service (QoS)[2] refers to the capability of a network to provide better service to selected network traffic over various technologies. The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Also important is making sure that providing priority for one or more flows does not make other flows fail. QoS technologies provide the elemental building blocks that will be used for future business applications in campus, WAN, and service provider networks. Almost any network can take advantage of QoS for optimum efficiency, whether it is a small corporate network, an Internet service provider, or an enterprise network. The QoS software provides these benefits:

- **Control over resources** - To have control over which resources (bandwidth, equipment, wide-area facilities, and so on) are being used. For example, to limit the bandwidth consumed over a backbone link by FTP transfers or give priority to an important database access.
- **More efficient use of network resources** - To know what your network is being used for and that you are servicing the most important traffic to your business.

- **Tailored services** - The control and visibility provided by QoS enables Internet service providers to offer carefully tailored grades of service differentiation to their customers.
- **Coexistence of mission-critical applications** - To make certain that your WAN is used efficiently by mission-critical applications that are most important to your business, that bandwidth and minimum delays required by time-sensitive multimedia and voice applications are available, and that other applications using the link get their fair service without interfering with mission-critical traffic.
- **Foundation for a fully integrated network in the future** - It is a good first step toward the fully integrated multimedia network needed in the near future.

2.1.1 Why IP QoS

- Application X is slow(not enough bandwidth) Lack of Bandwidth is caused by Multiple flows are contesting for a limited amount of bandwidth.
- Video broadcast occasionally stalls(delay temporarily increases (jitter) Variable delay is caused because Sometimes there is a lot of other traffic, which results in more delay.
- Phone calls over IP are no better than over satellite(too much delay) Too much delay is caused because Packets have to traverse many network devices and links.
- Phone calls can have very bad voice quality(too many phone calls (admission control)
- ATMs (the money-dispensing type) are nonresponsive(too many drops) Drops are caused when a link is congested.

2.1.1.1 How to Increase Available Bandwidth?

Take bandwidth from some less important applications.

2.1.1.2 How to Reduce delay?

Forward the important packets first.

FIGURE 2.1: How to Increase Available Bandwidth

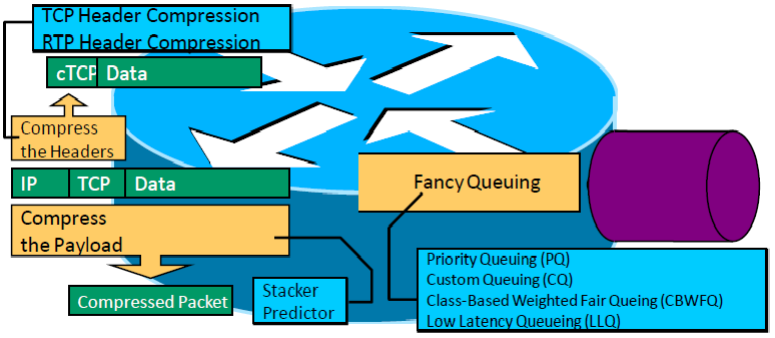


FIGURE 2.2: How to Reduce delay

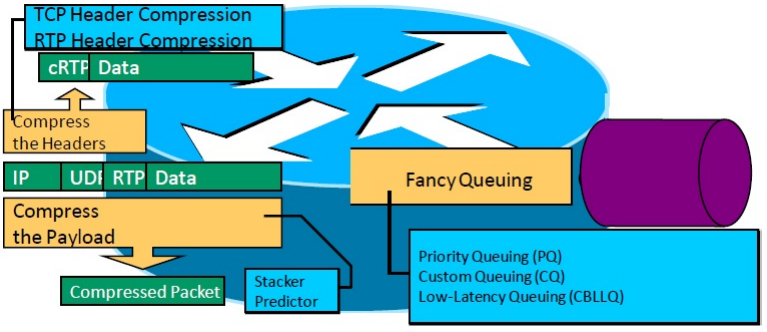
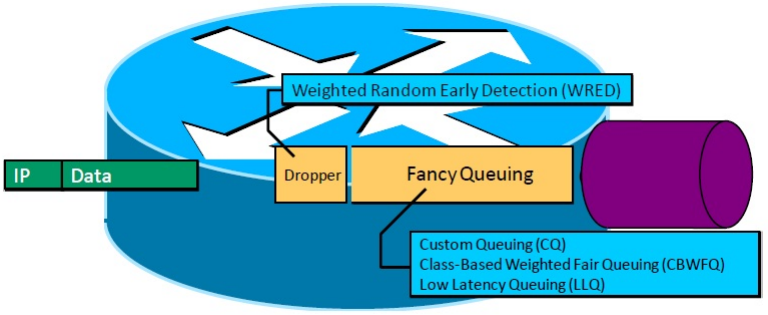


FIGURE 2.3: How to prevent packet loss



2.1.1.3 How to prevent packet loss?

Guarantee enough bandwidth to sensitive packets. Prevent congestion by randomly dropping less important packets before congestion occurs.

2.1.2 How Can QoS Be Applied?

- **Best effort** - no QoS is applied to packets (default behavior)

- **Integrated Services model** - applications signal to the network that they require special QoS
- **Differentiated Services model** - the network recognizes classes that require special QoS

Here we will be discussing Differentiated Services model of the QoS.

2.2 Basic QoS Architecture

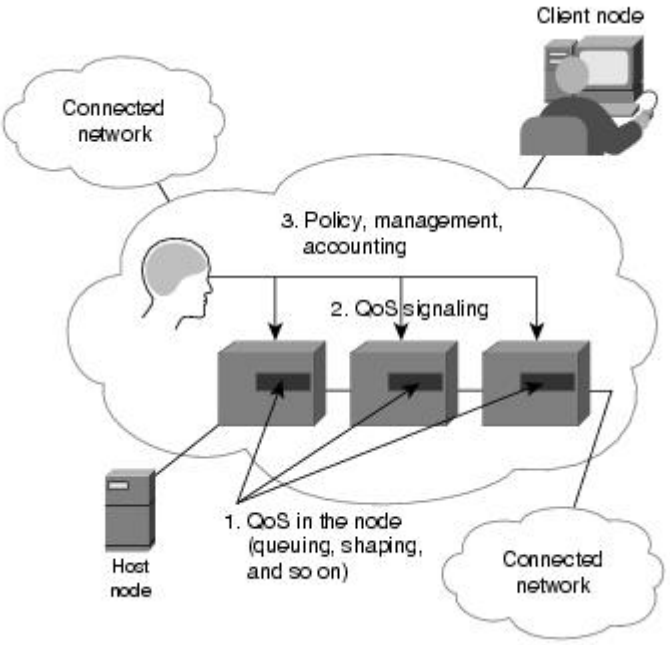
The basic architecture[6] introduces the three fundamental pieces for QoS implementation:

- QoS identification and marking techniques for coordinating QoS from end to end between network elements.
- QoS within a single network element (for example, queuing, scheduling, and traffic-shaping tools).
- QoS policy, management, and accounting functions to control and administer end-to-end traffic across a network.

2.2.1 QoS Identification and Marking

To provide priority to certain flows, the flow must first be identified and (if desired) marked. These two tasks are commonly referred to as just classification. Historically, identification was done using access control lists (ACLs). ACLs identify traffic for congestion-management tools such as PQ and CQ. Because PQ and CQ are placed on routers on a hop-by-hop basis (that is, priority settings for QoS pertain only to that router and are not passed to subsequent router hops in the network), identification of the packet is used only within a single router. In some instances, CBWFQ classification is for only a single router. This is contrasted by setting IP precedence bits. Features such as policy-based routing and committed access rate (CAR) can be used to set precedence based on extended access list classification. This allows considerable flexibility for precedence assignment, including assignment by application or user, by destination and

FIGURE 2.4: QoS Architecture



source subnet, and so on. Typically this functionality is deployed as close to the edge of the network (or administrative domain) as possible so that each subsequent network element can provide service based on the determined policy. Network-based application recognition (NBAR) is used to identify traffic more granularly. For example, URLs in an HTTP packet can be identified. Once the packet has been identified, it can be marked with a precedence setting.

2.2.1.1 Policy Setting with Policy-Based Routing

Policy-Based Routing (PBR) enables you to classify traffic based on extended access list criteria, set IP precedence bits, and even route to specific traffic-engineered paths that may be required to allow a specific QoS through the network. By setting precedence levels on incoming traffic and using them in combination with the queuing tools that will be described later in this report, you can create differentiated service. These tools provide powerful, simple, and flexible options for implementing QoS policies in your network.

Using policy-based routing, route maps are made to match on certain flow criteria and then set precedence bits when ACLs are matched.

The capability to set IP precedence bits should not be confused with PBR's primary capability: routing packets based on configured policies. Some applications or traffic

can benefit from QoS-specific routing-transferring stock records to a corporate office (for example, on a higher-bandwidth, higher-cost link for a short time), while transmitting routine application data such as e-mail over a lower-bandwidth, lower-cost link. PBR can be used to direct packets to take different paths than the path derived from the routing protocols. It provides a more flexible mechanism for routing packets, complementing the existing mechanisms provided by routing protocols.
(ACLs are used to control network access or to specify traffic for many features to act upon. An extended ACL is made up of one or more access control entries (ACEs). Each ACE specifies a source and destination for matching traffic.)

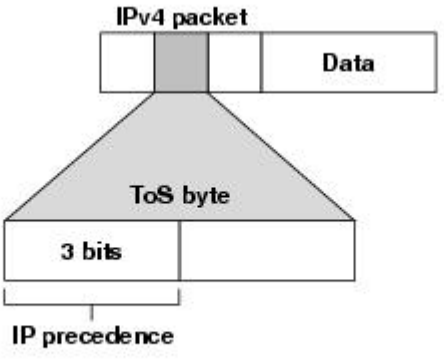
2.2.1.2 CAR: Setting IP Precedence

Similar in some ways to PBR, the CAR feature enables you to classify traffic on an incoming interface. It also allows specification of policies for handling traffic that exceeds a certain bandwidth allocation. CAR looks at traffic received on an interface, or a subset of that traffic selected by access list criteria, compares its rate to that of a configured token bucket, and then takes action based on the result (for example, drop or rewrite IP precedence). There is some confusion with using CAR to set IP precedence bits. An attempt to clear up any confusion follows. CAR (as its name describes) is used to police traffic flows to a committed access rate. CAR does this with a token bucket. A token bucket is a bucket with tokens in it that represent bytes (1 token = 1 byte). The bucket is filled with tokens at a user-configured rate. As packets arrive to be delivered, the system checks the bucket for tokens. If there are enough tokens in the bucket to match the size of the packet, those tokens are removed and the packet is passed (this packet conforms). If there aren't enough tokens, the packet is dropped (this packet exceeds). There are more options than just pass or drop. One option is to set the IP precedence bits. When the conform and exceed actions both say to set precedence bits to the same setting, then it is no longer a policing feature, but merely a method of setting IP precedence bits.

2.2.1.3 NBAR: Dynamic Identification of Flows

NBAR takes the identification portion of classification to another level. Looking deeper into the packet, identification can be performed, for example, to the URL or MIME type

FIGURE 2.5: This Diagram Shows the IP Precedence ToS Field in an IP Packet Header



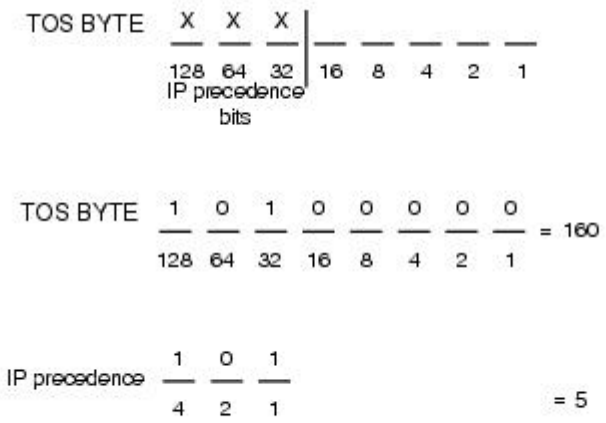
of an HTTP packet. This becomes essential as more applications become web-based. You would need to differentiate between an order being placed and casual web browsing. In addition, NBAR can identify various applications that use ephemeral ports. NBAR does this by looking at control packets to determine which ports the application decides to pass data on.

NBAR adds a couple of interesting features that make it extremely valuable. One feature is a protocol discovery capability. This allows NBAR to baseline the protocols on an interface. NBAR lists the protocols that it can identify and provides statistics on each one. Another feature is the Packet Description Language Module (PDLM), which allows additional protocols to be easily added to NBAR's list of identifiable protocols. These modules are created and loaded into Flash memory, which then is uploaded into RAM. Using PDLMs, additional protocols can be added to the list without upgrading the IOS level or rebooting the router.

2.2.1.4 IP Precedence: Differentiated QoS

IP precedence utilizes the 3 precedence bits in the IPv4 header's Type of Service (ToS) field to specify class of service for each packet, as shown in Figure: This Diagram Shows the IP Precedence ToS Field in an IP Packet Header. You can partition traffic in up to six classes of service using IP precedence (two others are reserved for internal network use). The queuing technologies throughout the network can then use this signal to provide the appropriate expedited handling. The 3 most significant bits (correlating to binary settings 32, 64, and 128) of the Type of Service (ToS) field in the IP header constitute the bits used for IP precedence. These bits are used to provide a priority from

FIGURE 2.6: IP Precedence



0 to 7 (settings of 6 and 7 are reserved and are not to be set by a network administrator) for the IP packet.

Because only 3 bits of the ToS byte are used for IP precedence, you need to differentiate these bits from the rest of the ToS byte. In Figure: IP Precedence, a 1 in the first and third bit positions (viewing from left to right) correlates to an IP precedence setting of 5, but when viewing the ToS byte in a Sniffer trace, it will show 160. You need to be able to translate these settings. Traffic that is identified can be marked by setting the IP precedence bits. Thus, it needs to be classified only once.

2.2.2 Congestion-Management Tools

One way network elements handle an overflow of arriving traffic is to use a queuing algorithm to sort the traffic, and then determine some method of prioritizing it onto an output link. This includes the following queuing tools:

- First-in, first-out (FIFO) queuing
- Priority queuing (PQ)
- Custom queuing (CQ)
- Flow-based weighted fair queuing (WFQ)
- Class-based weighted fair queuing (CBWFQ)

Each queuing algorithm was designed to solve a specific network traffic problem and has a particular effect on network performance, as described in the following sections.
In this project, we are modeling the queue dynamics in PQ and so we will concentrate on that more compared to the other queuing methods.

2.2.2.1 FIFO: Basic Store-and-Forward Capability

In its simplest form, FIFO queuing involves storing packets when the network is congested and forwarding them in order of arrival when the network is no longer congested. FIFO is the default queuing algorithm in some instances, thus requiring no configuration, but it has several shortcomings.
Most importantly, FIFO queuing makes no decision about packet priority; the order of arrival determines bandwidth, promptness, and buffer allocation. Nor does it provide protection against ill-behaved applications (sources). Bursty sources can cause long delays in delivering time-sensitive application traffic, and potentially to network control and signaling messages. FIFO queuing was a necessary first step in controlling network traffic, but today's intelligent networks need more sophisticated algorithms. In addition, a full queue causes tail drops. This is undesirable because the packet dropped could have been a high-priority packet. The router couldn't prevent this packet from being dropped because there was no room in the queue for it (in addition to the fact that FIFO cannot tell a high-priority packet from a low-priority packet).

2.2.2.2 PQ: Prioritizing Traffic

PQ ensures that important traffic gets the fastest handling at each point where it is used. It was designed to give strict priority to important traffic. Priority queuing can flexibly prioritize according to network protocol, incoming interface, packet size, source/destination address, and so on. In PQ, each packet is placed in one of four queues-high, medium, normal, or low-based on an assigned priority. Packets that are not classified by this priority list mechanism fall into the normal queue (see Figure: Priority Queuing Places Data into Four Levels of Queues: High, Medium, Normal, and Low). During transmission, the algorithm gives higher-priority queues absolute preferential treatment over low-priority queues.
PQ is useful for making sure that mission-critical traffic traversing various WAN links

FIGURE 2.7: Priority Queuing Places Data into Four Levels of Queues: High, Medium, Normal, and Low

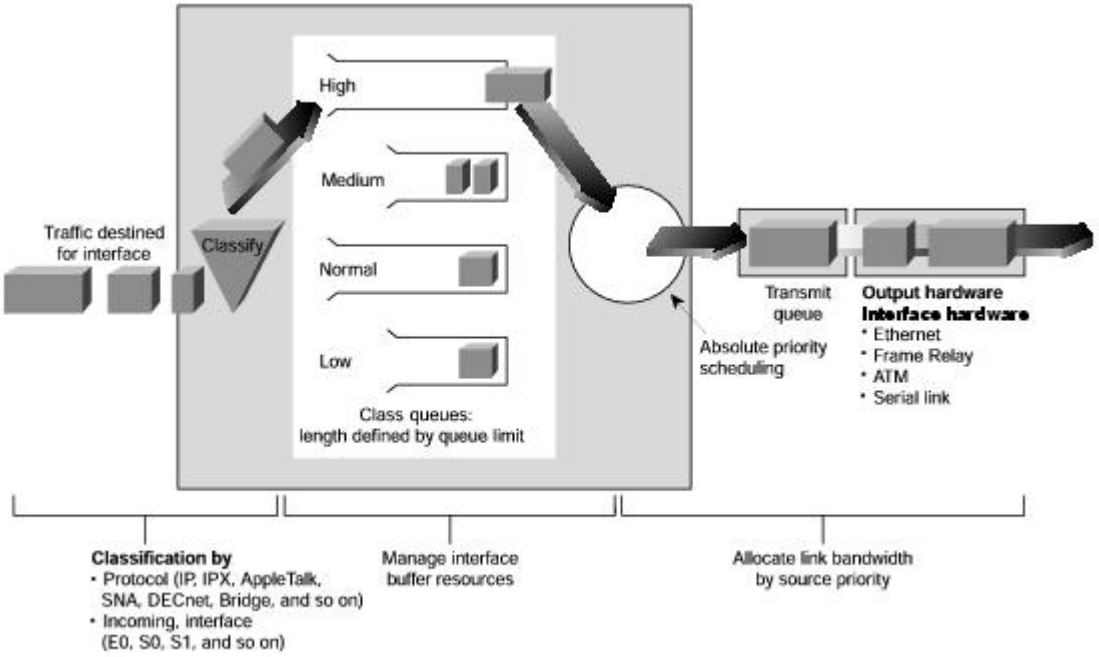
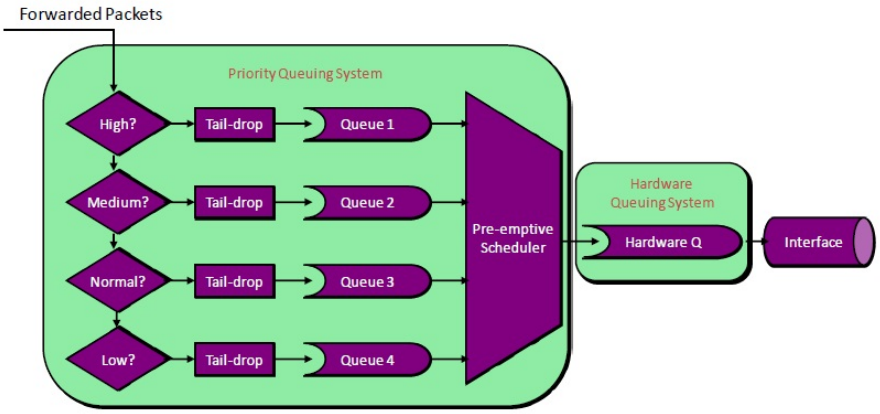


FIGURE 2.8: Priority Queuing in QoS

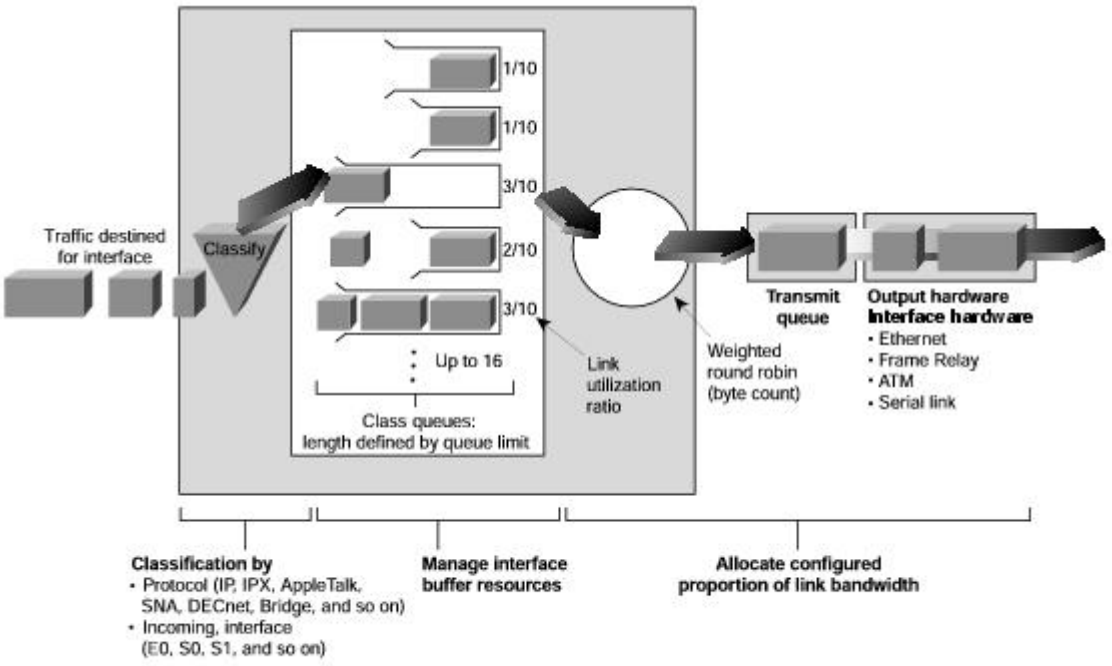


gets priority treatment. PQ currently uses static configuration and thus does not automatically adapt to changing network requirements. The queuing is explained even better with the figure 2.8.

2.2.2.3 CQ: Guaranteeing Bandwidth

CQ was designed to allow various applications or organizations to share the network among applications with specific minimum bandwidth or latency requirements. In these

FIGURE 2.9: Custom Queuing



environments, bandwidth must be shared proportionally between applications and users. This feature is used to provide guaranteed bandwidth at a potential congestion point, ensuring the specified traffic a fixed portion of available bandwidth and leaving the remaining bandwidth to other traffic. Custom queuing handles traffic by assigning a specified amount of queue space to each class of packets and then servicing the queues in a round-robin fashion (see Figure: Custom Queuing). The queuing algorithm places the messages in one of 17 queues (queue 0 holds system messages such as keepalives, signaling, and so on) and is emptied with weighted priority. The router services queues 1 through 16 in round-robin order, dequeuing a configured byte count from each queue in each cycle. This feature ensures that no application (or specified group of applications) achieves more than a predetermined proportion of overall capacity when the line is under stress. Like PQ, CQ is statically configured and does not automatically adapt to changing network conditions.

2.2.2.4 Flow-Based WFQ: Creating Fairness Among Flows

For situations in which it is desirable to provide consistent response time to heavy and light network users alike without adding excessive bandwidth, the solution is flow-based

WFQ (commonly referred to as just WFQ). It is a flow-based queuing algorithm that creates bit-wise fairness by allowing each queue to be serviced fairly in terms of byte count. For example, if queue 1 has 100-byte packets and queue 2 has 50-byte packets, the WFQ algorithm will take two packets from queue 2 for every one packet from queue 1. This makes service fair for each queue: 100 bytes each time the queue is serviced.

WFQ ensures that queues do not starve for bandwidth and that traffic gets predictable service. Low-volume traffic streams-which comprise the majority of traffic-receive increased service, transmitting the same number of bytes as high-volume streams. This behavior results in what appears to be preferential treatment for low-volume traffic, when in actuality it is creating fairness, as shown in Figure: WFQ.

WFQ is designed to minimize configuration effort, and it automatically adapts to changing network traffic conditions.Flow-based WFQ creates flows based on a number of characteristics in a packet. Each flow (also referred to as a conversation) is given its own queue for buffering if congestion is experienced.

The weighted portion of WFQ comes from the use of IP precedence bits to provide greater service for certain queues. Using settings 0 to 5 (6 and 7 are reserved), WFQ uses its algorithm to determine how much more service to provide to a queue.

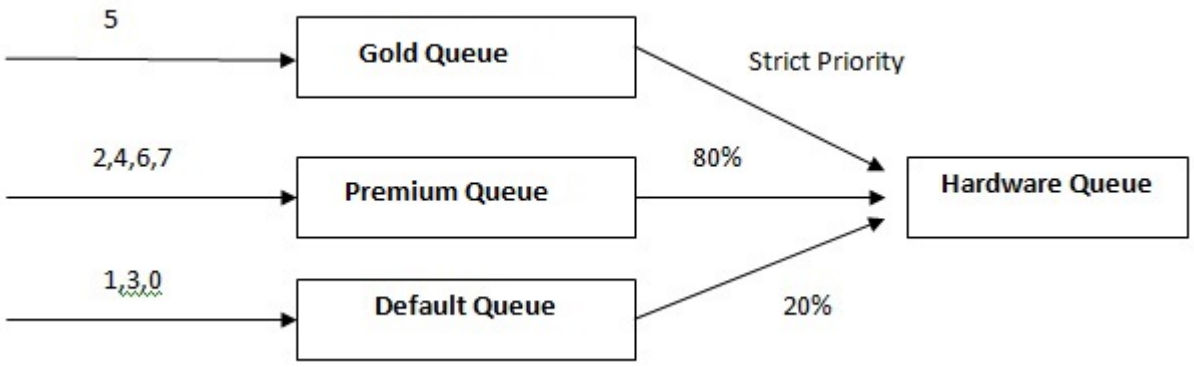
A weight is a number calculated from the IP precedence setting for a packet in flow. This weight is used in WFQ's algorithm to determine when the packet will be serviced. Weight = (4096 / (IP precedence + 1) Weight = (32384 / (IP precedence + 1) The numerator of the equation changed from 4096 to 32384 in a v12.0 maintenance release. Weight settings can be viewed using the show queue ¡interface¿ command.

Effect of IP Precedence Settings The effect of IP precedence settings is described here: If you have one flow at each precedence level on an interface, each flow will get precedence + 1 parts of the link, as follows: 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36

The flows will get 8/36, 7/36, 6/36, and 5/36 of the link, and so on. However, if you have 18 precedence-1 flow and 1 of each of the others-the formula looks like this: 1 + 1 8 * 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36 - 2 + 18 * 2 = 70

The flows will get 8/70, 7/70, 6/70, 5/70, 4/70, 3/70, 2/70, and 1/70 of the link, and 18 of the flows will each get approximately 2/70 of the link.

FIGURE 2.10: Modified Deficit Round Robin



2.2.2.5 Class-Based WFQ: Ensuring Network Bandwidth

When you want to provide a minimum amount of bandwidth, use CBWFQ. This is in comparison to a desire to provide a maximum amount of bandwidth. CAR and traffic shaping are used in that case.

CBWFQ allows a network administrator to create minimum guaranteed bandwidth classes. Instead of providing a queue for each individual flow, a class is defined that consists of one or more flows. Each class can be guaranteed a minimum amount of bandwidth.

2.2.2.6 MDRR (Modified Deficit Round Robin)

Here we will describe the congestion management mechanism implemented in Telecom Italia.

The input flows are classified based on IP Precedence and served through three queues-High,Medium and Default.

The High Queue is given strict high priority and is pre-empted with Medium and Default queues.When the High Queue is empty the medium and default queues are served in round robin fashion and each queue is reserved with a percentage of link bandwidth.Each queue has a maximum number of bytes it can hold. From the figure 2.10 it is seen that medium queue is given 80% of the available link bandwidth when the high queue is empty and the default queue is given 20% of the available link bandwidth.

2.2.3 Queue Management (Congestion-Avoidance Tools)

Congestion avoidance is a form of queue management. Congestion-avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottlenecks, as opposed to congestion-management techniques that operate to control congestion after it occurs.

2.2.3.1 WRED: Avoiding Congestion

The random early detection (RED) algorithms are designed to avoid congestion in internetworks before it becomes a problem. RED works by monitoring traffic load at points in the network and stochastically discarding packets if the congestion begins to increase. The result of the drop is that the source detects the dropped traffic and slows its transmission. RED is primarily designed to work with TCP in IP internetwork environments.

WRED Cooperation with QoS Signaling Technologies: WRED combines the capabilities of the RED algorithm with IP precedence. This combination provides for preferential traffic handling for higher-priority packets. It can selectively discard lower-priority traffic when the interface starts to get congested and can provide differentiated performance characteristics for different classes of service. Within each queue, a finite number of packets can be housed. A full queue causes tail drops. Tail drops are dropped packets that could not fit into the queue because the queue was full. This is undesirable because the packet discarded may have been a high-priority packet and the router did not have a chance to queue it. If the queue is not full, the router can look at the priority of all arriving packets and drop the lower-priority packets, allowing high-priority packets into the queue. Through managing the depth of the queue (the number of packets in the queue) by dropping various packets, the router can do its best to make sure that the queue does not fill and that tail drops are not experienced. This allows the router to make the decision on which packets get dropped when the queue depth increases. WRED also helps prevent overall congestion in an internetwork. WRED uses a minimum threshold for each IP precedence level to determine when a packet can be dropped. (The minimum threshold must be exceeded for WRED to consider a packet as a candidate for being dropped.)

2.2.3.2 Flow RED: RED for Non-TCP-Compliant Flows

WRED is primarily used for TCP flows that will scale back transmission if a packet is dropped. There are non-TCP-compliant flows that do not scale back when packets are dropped. Flow RED is used to deal with such flows. The approach is to increase the probability of dropping a flow if it exceeds a threshold. Flow-based WRED relies on these two main approaches to remedy the problem of linear packet dumping:

- It classifies incoming traffic into flows based on parameters such as destination and source addresses and ports.
- It maintains state about active flows, which are flows that have packets in the output queues.

Flow-based WRED uses this classification and state information to ensure that each flow does not consume more than its permitted share of the output buffer resources. Flow-based WRED determines which flows monopolize resources, and it more heavily penalizes these flows. This is how flow-based WRED ensures fairness among flows: It maintains a count of the number of active flows that exist through an output interface. Given the number of active flows and the output queue size, flow-based WRED determines the number of buffers available per flow. To allow for some burstiness, flow-based WRED scales the number of buffers available per flow by a configured factor and allows each active flow to have a certain number of packets in the output queue. This scaling factor is common to all flows. The outcome of the scaled number of buffers becomes the per-flow limit. When a flow exceeds the per-flow limit, the probability that a packet from that flow will be dropped increases.

2.2.4 Traffic-Shaping and Policing Tools

2.2.4.1 CAR: Managing Access Bandwidth Policy and Performing Policing

As described earlier, fundamentally, QoS provides priority either by raising the priority of one flow or by limiting the priority of another. CAR is used to limit the bandwidth of a flow in order to favor another flow. In the "Classification" section, earlier in this

chapter, a generic token bucket was described. In that description, packets that conform are passed, and packets that exceed are dropped. A number of actions can be performed. These actions consist of transmitting, dropping, setting IP precedence bits, and continuing (this refers to cascading CAR statements). This flexibility allows for a number of ways to act upon traffic.

2.2.4.2 GTS: Controlling Outbound Traffic Flow

GTS provides a mechanism to control the traffic flow on a particular interface. It reduces outbound traffic flow to avoid congestion by constraining specified traffic to a particular bit rate (it also uses a token bucket approach) while queuing bursts of the specified traffic. So, any traffic above the configured rate is queued. This differs from CAR, in which packets are not queued.

2.2.5 Link Efficiency Mechanisms

Link efficiency mechanisms-link fragmentation and interleaving (LFI) and real-time protocol header compression (RTP-HC)-which work with queuing and traffic shaping to improve the efficiency and predictability of the application service levels.

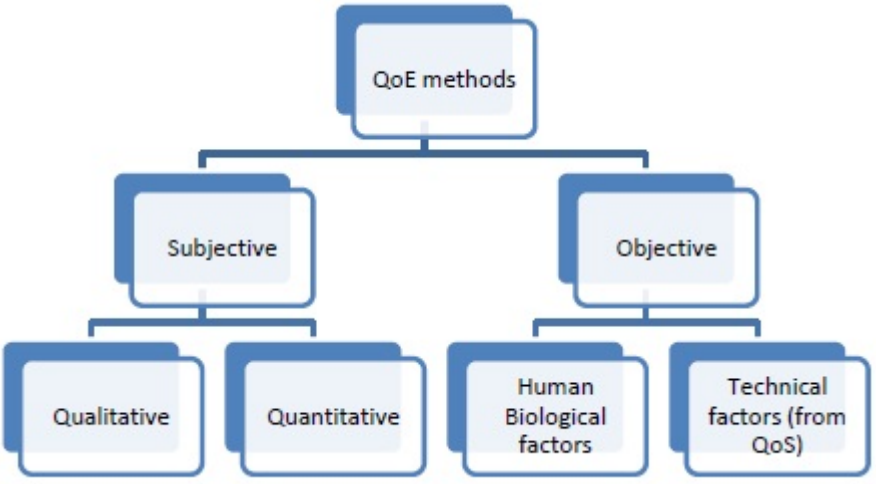
2.2.5.1 LFI: Fragmenting and Interleaving IP Traffic

LFI feature reduces delay and jitter on slower-speed links by breaking up large datagrams and interleaving low-delay traffic packets with the resulting smaller packets.

2.2.5.2 RTP Header Compression: Increasing Efficiency of Real-Time Traffic

Real-Time Transport Protocol header compression reduces line overhead for multimedia Real-Time Transport Protocol traffic with a corresponding reduction in delay, especially for traffic that uses short packets relative to header length.

FIGURE 2.11: Different ways of measuring QoE



2.3 Introduction to QoE

Quality-of-Experience (QoE) is a human centric notion that produces the blue print of human perception, feelings, needs and intentions while Quality-of-Service (QoS) is a technology centric metric used to assess the performance of a multimedia application and/or network.QoE is usually expressed using a Mean Opinion Score (MOS) of 1 ("Bad") to 5 ("Excellent"). In particular we are interested in QoE for video applications.To ensure superior video QoE, it is important to understand the relationship between QoE and QoS.There are different ways of estimating QoE from QoS. Following are some of the estimation techniques that are obtained from the references [8],[11],[7].

2.3.1 QoE from Network QoS

In this method, QoE is estimated using subjective analysis with QoS parameters.Experiments were conducted with users, with different QoS parameters and relationship between QoS and average PVQ (perceived video quality, a QoE measure) were explored.The QoS parameters that were used for the experiment are Packet loss Rate, Packet Reorder Rate and Video Bit rate. Since we cannot measure packet reorder using our model, we will concentrate on packet loss rate and video bit rate for measuring QoE.The figure 2.12 shows the relationship between QoS parameters and QoE after conducting experiments with users.

FIGURE 2.12: PVQ Vs Packet loss rate % and Video Bit Rate

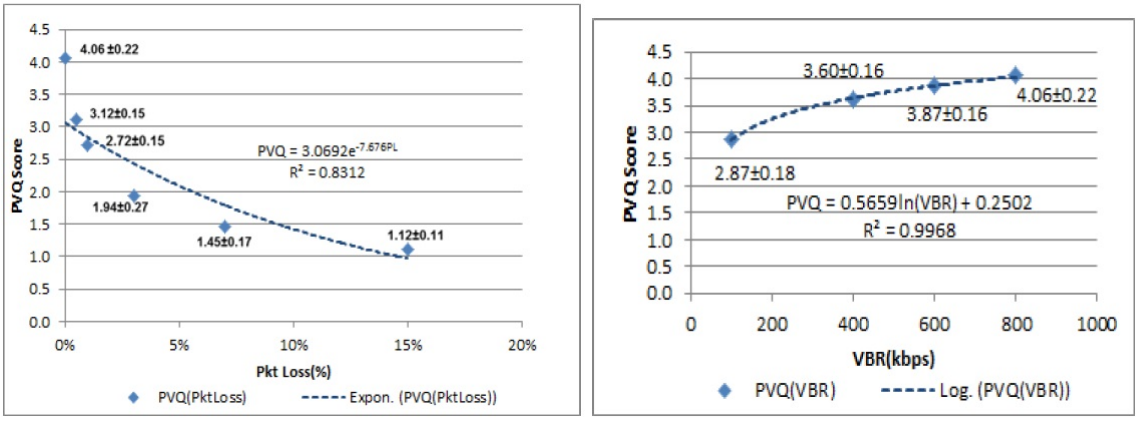
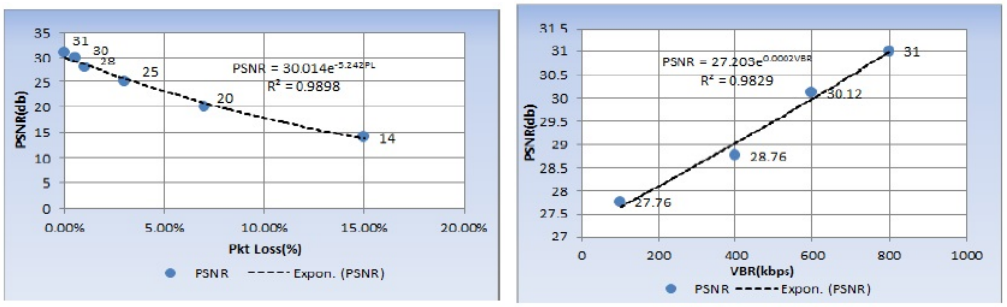


FIGURE 2.13: PSNR Vs Packet loss rate % and Video Bit Rate



2.3.2 QoE from PSNR(Peak Signal to Noise Ratio)

PSNR stands for Peak Signal to Noise Ratio and it can be calculated by Full Reference Objective Method.It is based on the mathematical difference between every pixel of the processed video and the original video. The relationships between PSNR and three QoS parameters namely packet loss rate, Video Bit rate and packet reorder rate. Since the packet reorder rate cannot be measured with our model, we will concentrate on the other two QoS parameters.PSNR scores followed somewhat similar trends as those observed with subjective PVQ ratings, with the exception of video bit rates, which showed a quasi-linear fitting, as opposed to a logarithmic fitting with PVQ.The figures 2.13 depicts the relationships. PSNR and PVQ follow an exponential relationship and it is depicted in the figure 2.14.

2.3.3 Network QoS-¿Application QoS-¿QoE

In this approach, analytical models are used to estimate Application QoS from Network QoS and subjective experiments with users were conducted to evaluate QoE. The

FIGURE 2.14: PVQ Vs PSNR

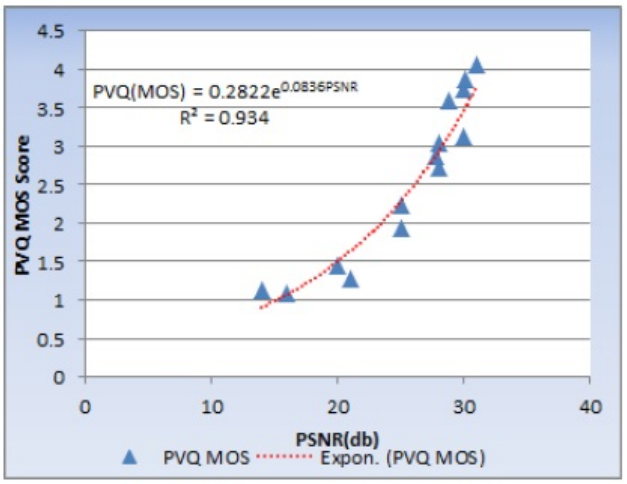
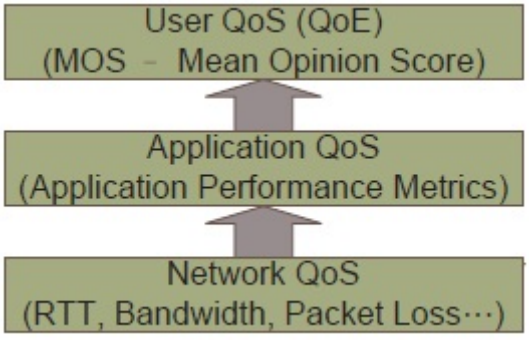


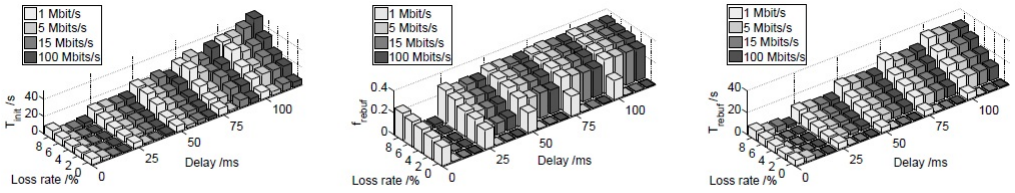
FIGURE 2.15: Three levels of QoS



reference [11] performed analytical studies to model the relationship between network QoS and application QoS.In particular, we use a set of application performance metrics (APM) for the study: (1) Initial buffering time, (2) mean duration of a rebuffering event, and (3) rebuffering frequency.

- **Initial buffering time** (denoted by T_{init}): This metric measures the period between the starting time of loading a video and the starting time of playing it.
- **Mean rebuffering duration** (denoted by T_{rebuf}): This metric measures the average duration of a rebuffering event.
- **Rebuffering frequency** (denoted by f_{rebuf}): When the amount of buffered video data decreases to a low value,the playback will pause, and the player will enter

FIGURE 2.16: Three APMs with different network quality



(a) T_{init}. (b) f_{rebuf}. (c) T_{rebuf}.

FIGURE 2.17: Three Levels of APMs

Level	APMs		
	T_{init}	f_{rebuf}	T_{rebuf}
Low	$0-1$ seconds	$0-0.02$	$0-5$ seconds
Medium	$1-5$ seconds	$0.02-0.15$	$5-10$ seconds
High	>5 seconds	>0.15	>10 seconds

into a rebuffering state. This metric measures how frequent the rebuffering events occur.

The QoS parameters used for estimating application QoS are loss rate %, delay and network bandwidth.The figure 2.16 shows the three Application Performance Metrics with under different network path quality. The application performance metrics are classified into three levels as shown in the figure 2.17. A regression analysis was done to acquire a relationship between QoE and application QoS. As shown in Equation below, the coefficients of the three APMs are all negative, thus a higher level of APMs giving a lower MOS.

$$\text{MOS} = 4.23 - 0.0672 * L_{ti} - 0.742 * L_{fr} - 0.106 * L_{tr}$$

where L_{ti}, L_{fr} and L_{tr} are the respective levels of T_{init},f_{rebuf} and T_{rebuf} . We use 1, 2, and 3 to represent the "low","medium", and "high" levels, respectively.

2.3.4 Other Methods for estimating QoE

There are other methods to measure QoE for video applications.

FIGURE 2.18: Negative Impact Vs Stalling Parameters

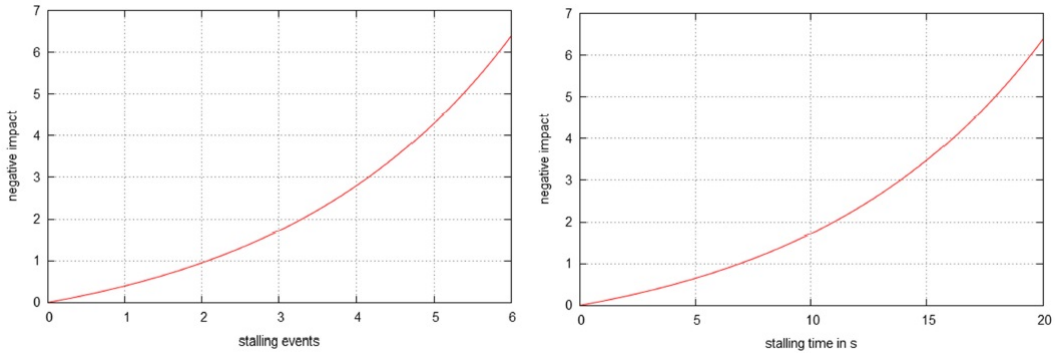
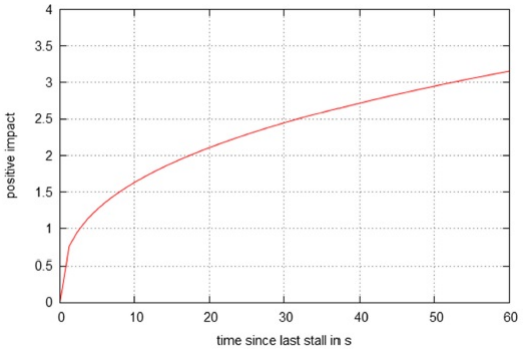


FIGURE 2.19: Positive Impact Vs Stalling Parameters



- **QMON** - Quality Monitoring (QMON) mechanism performs a playout buffer fill level estimation based on TCP flow observation in the measurement point and does a QoE evaluation based on the number and duration of occurred stalling events.QoE estimation relies on the occurrence of stalling events only. Three criteria are vital for the calculation. The first one is the number of occurred stalling events, the second one is the duration of the corresponding stall and the third one is the time since the last stall finished. The first two parameters are merged into a so called "Negative Impact" (NI) and the third parameter is merged into "Positive Impact"(PI). For each video a basic MOS of 4.5 is assumed and is decreased by the calculated NI at each point in time. The resulting equation is shown below. MOS=4.5-NI+PI.

- **Machine learning techniques** - Using machine learning techniques (K-nearest neighbour algorithm) to discern the connection between the objective data and video QoE.At first,historical data about video streams consisting of stream data and quality ratings were collected.This historical data forms the training set for

the machine learning algorithm.During the training phase, two heuristics for the algorithm were determined: how many neighbors (closest to the stream to rate) to use when assigning a quality rating (K), and how far forwards and backwards in time to shift when measuring the distance between two streams.The values, K and distance were used to estimate the QoE for unrated video streams.

Chapter 3

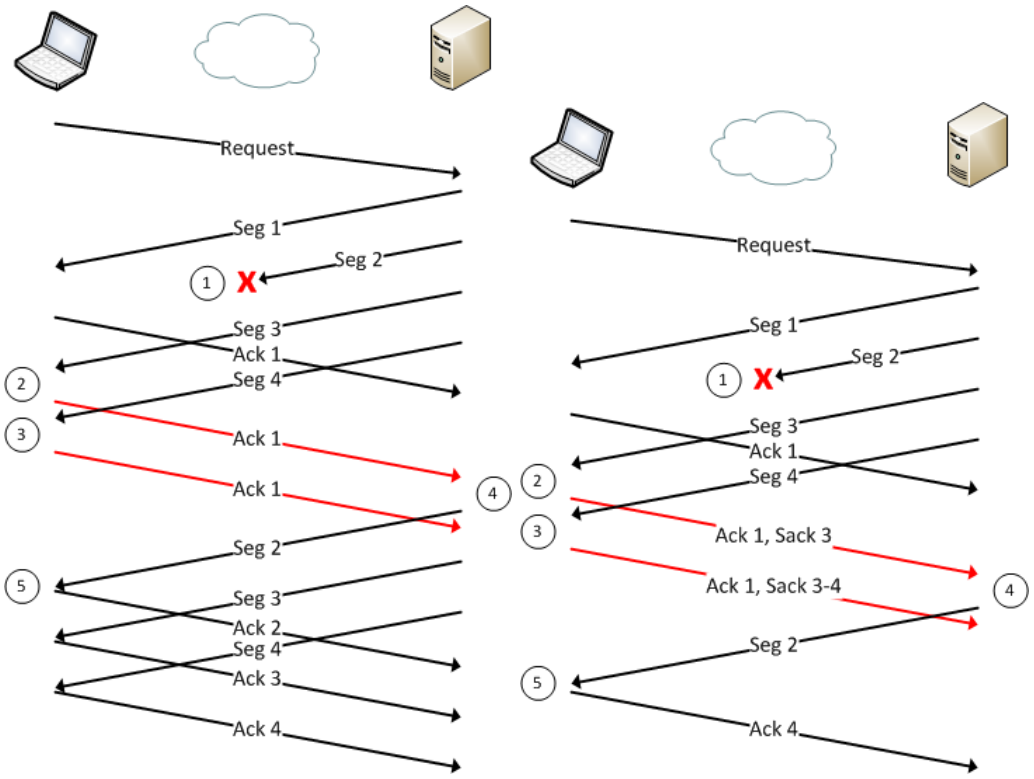
Hybrid Models for TCP-SACK,Queue Dynamics, Priority Queuing and MDRR

3.1 TCP Sack Hybrid model

In this section, a hybrid model for a single TCP flow f according to the TCP SACK algorithm is described.

TCP uses sequence and acknowledgment numbers to keep track of data in bidirectional transit between two end hosts. In an ideal TCP stream, all packets are delivered to either end successfully, and in order.The left side of figure 3.1 illustrates a TCP connection taking place between a client and server separated by a network. Time progresses vertically from top to bottom as packets are sent. The client sends some request to the server, and the server formulates a response broken into four TCP segments (packets). The server transmits all four packets in response to the request.The second response packet is dropped somewhere on the network and never reaches the host.The client receives segment 3. Upon examining the segment's sequence number, the client realizes this segment is out of order; there is data missing between the last segment received and this one. The client transmits a duplicate acknowledgment for packet 1 to alert the server that it has not received any (reliable) data beyond packet 1.As the server is not yet aware that

FIGURE 3.1: TCP and TCP-SACK



anything is wrong (because it has not yet received the client's duplicate acknowledgment), it continues by sending segment 4. The client realizes that it is still missing data, and repeats its behavior in step three by sending another duplicate acknowledgment for packet 1.The server receives the client's first duplicate acknowledgment for packet 1. Because the client has only confirmed receipt of the first of the four segments, the server must re-transmit all three remaining segments in the response.The second duplicate acknowledgment received from the client is ignored.Although only packet 1 was lost, the server was required to retransmit packets 3 and 4 as well, because the client had no way to confirm that it had received those packets.By introducing the selective acknowledgment (SACK) TCP option,a TCP option is appended to the duplicate acknowledgement containing a range of non contiguous data received.In other words, it allows the client to say "I only have up to packet 1 in order, but I also have received packets 3 and 4". This allows the server to retransmit only the packet(s) that were not received by the client.Support for SACK is negotiated at the beginning of a TCP connection; if both hosts support it, it may be used.

The right side figure 1.3 shows the TCP connection between client and server with SACK enabled. When response for segment 1 is lost,the client realizes it is missing a segment

between segments 1 and 3. It sends a duplicate acknowledgment for segment 1, and attaches a SACK option indicating that it has received segment 3.The client receives segment 4 and sends another duplicate acknowledgment for segment 1, but this time expands the SACK option to show that it has received segments 3 through 4.From this, the server deduces that the client is missing segment 1, so segment 1 is retransmitted.The client receives segment 2 and sends an acknowledgment to indicate that it has received all data up to an including segment 4.
The operating modes of TCP-SACK and the associated dynamics of data flow are given below.

3.1.1 Slow Start Mode

During this mode, the congestion window w_f is multiplied by 2 every RTT:

$$\dot{w_f} = \frac{\log(2)}{RTT_f} w_f, \tag{3.1}$$

If RTT_f was constant, we get

$$w_f(t + RTT_f) = \mathrm{e}^{\log(2) \int_t^{t+RTT_f} \frac{1}{RTT_f} \mathrm{d}\tau} w_f(t) \approx 2w_f(t) \tag{3.2}$$

Since w_f packets are sent each round trip time, the instantaneous sending rate r_f should be given by

$$r_f = \frac{w_f}{RTT_f} \tag{3.3}$$

However this formula needs to be corrected to

$$r_f = \frac{\beta w_f}{RTT_f} \tag{3.4}$$

because slow start packets are sent in bursts. The slow start mode lasts until a drop or a timeout are detected. Detection of a drop leads the system in a fast recovery mode, whereas a timeout leads the system in a timeout mode. When w_f exceeds receiver's advertised window size w^{adv}, the sending rate is limited by w^{adv} and we can replace the sending rate r_f by

$$r_f = \frac{min\{w_f, w^{adv}\}}{RTT_f} \tag{3.5}$$

When the congestion window reaches w^{adv}, the system leaves the slow-start mode and goes to teh congestion-avoidance mode.

3.1.2 Congestion Avoidance Mode

During this mode we have a linear increase of the congestion window size, with an increase equal to the packet size L for each RTT:

$$\dot{w_f} = \frac{L}{RTT_f}, \tag{3.6}$$

with the instantaneous sending rate given by (1.12). When w^{adv} is finite, the equation (1.12) should be replaced by

$$r_f = \frac{min\{w_f, w^{adv}\}}{RTT_f} \tag{3.7}$$

The congestion avoidance mode lasts until a drop or timeout are detected. Detection of a drop leads the system to a fast-recovery mode, whereas detection of timeout leads to the time-out mode.

3.1.3 Fast Recovery Mode

This mode is visited when a drop is detected, which occurs some time after the drop actually occurs. When a single drop occurs, the sender leaves this mode when an acknowledgement arrives. When multiple drops occur, the sender leaves this mode depending on the TCP model.

Here we consider the TCP-Sack algorithm: when n_{drop} drops occur, the sender will attempt to retransmit all these packet as soon as the congestion window allows it. The first packet dropped is retransmitted immediately and the congestion window is dived by two. Then, for each received acknowledgement the congestion window is increased by 1. However, until the first retransmission succeeds, the number of outstanding packets is not decreased when acks arrive.Suppose now that a drop was detected at time t_0 and we have the value of the congestion window right before the drop $w_f(t_0^-)$.During the first RTT after the retransmission (i.e. from t_0 to $t_0 + RTT_f$) the number of outstanding packets is w_f (t_0^-); the number of duplicate received acknowledgements is equal to $w_f(t_0^-$)- n_{drop} (in particular we consider 3 duplicate acknowledgment that trigger the retransmission), and a single non-duplicate acknowledgement is received (corresponding

to successful retransmission). The total number of packets sent during this interval will be one, plus the number of duplicate acknowledgements received, minus $w_f\ (t_0^-\ )/2$. We have to subtract $w_f\ (t_0^-\ )/2$ because the first $w_f\ (t_0^-\ )/2$ received acknowledgements will increase the congestion window up to the number of outstanding packets but will not lead to transmission because the congestion window is still below the number of outstanding packets. This leads to a total of $1 + w_f\ (t_0^-\ )/2$ - n_{drop} packets sent. So we can say that the averaging sending rate can be modeled by

$$r_f = \frac{1 + w_f(t_0^-)/2 - n_{drop}}{RTT_f} on[t_0, t_0 + RTT_f]. \tag{3.8}$$

In case a single packet was dropped fast recovery will terminate at $t_0 + RTT_f$, otherwise it will continue until all the retransmissions take place successfully. However, starting from t_0 +RTT on, each received acknowledgement will also decrease the number of outstanding packets: as a consequence we have an exponential increase in the window size. In particular, from $t_0 + RTT_f$ to $t_0 + 2RTT_f$, the number of received acknowledgments is $1 + w_f\ (t_0^-)/2$ -n_{drop} (i.e. the number of packets sent in the previous interval) and each will both increase the congestion window size and decrease the number of outstanding packets. Thus,

$$r_f = \frac{2(1 + w_f(t_0^-)/2 - n_{drop})}{RTT_f} on[t_0 + RTT_f, t_0 + 2RTT_f]. \tag{3.9}$$

On each subsequent interval, the sending rate increases exponentially until the total n_{drop} packets are successfully retransmitted. In k round trip times the total number of packets retransmitted is equal to

$$\sum_{i=0}^{k-1} 2^i (1 + w_f(t_0^-)/2 - n_{drop}) = (2^k - 1)(1 + w_f(t_0^-)/2 - n_{drop}) \tag{3.10}$$

and the sender will exit fast recovery when this number reaches n_{drop} value, i.e. when

$$(2^k - 1)(1 + w_f(t_0^-)/2 - n_{drop}) = n_{drop} \Longleftrightarrow k = \log_2 \frac{1 + w_f(t_0^-)/2}{1 + w_f(t_0^-)/2 - n_{drop}} \tag{3.11}$$

We can conclude that the hybrid model should remain in the fast recovery mode for approximately

$$n(w_f(t_0^-), n_{drop}) = [\log_2 \frac{1 + w_f(t_0^-)/2}{1 + w_f(t_0^-)/2 - n_{drop}}] \tag{3.12}$$

3.1.4 Timeout mode

Timeouts occur when the timeout timer exceeds a threshold that provides a measure of the present RTT.This timer is reset to zero whenever the sender receives an acknowledgement for a new packet. Even when there are drops, this should occur at least once every RTT_f , except in two particular cases: (1) $n_{drop} \geq w_f$-2, i.e. the number of duplicate acknowledgement is smaller or equal to 2, which are not enough to trigger a transition to the fast-recovery mode; (2) n_{drop} is sufficiently large so that the sender will not be able to exit fast recovery because it does not receive enough acknowledgments to retransmit all the dropped packets. As seen above, this particular condition corresponds to $n_{drop} \geq w_f/2 + 2$. These two cases can be combined into the following condition under which a timeout will occur:

$$w_f \leq max2 + n_{drop}, 2n_{drop} - 4$$

We also define the variable $ssthr_f$ that is set equal to half the congestion window size before a timeout occurs, whereas the congestion window after a timeout will remain 1, i.e.

$$ssthr_f = w_f(t_0^-)/2, w_f(t_0) = 1$$

with t_0 when a timeout occurs. At this point, and until w reaches $sstrh_f$, we have a multiplicative increase similar to the slow start mode. This mode lasts until w_f reaches $ssthr_f(t_0)$ or a drop/timeout occurs. The former leads to a transition into the congestion avoidance mode, whereas the latter to fast recovery mode/timeout mode.

3.1.5 Hybrid Model for TCP-Sack

The TCP-Sack algorithm is modeled by a hybrid system characterized by the 4 modes described above. Note that it takes into account that there is a delay between the occurrence of a drop and its detection. This drop-detection-delay is determined by the round trip time for the queue l where the drop occurred, i.e.:

$$DDD_f^l = \sum_{l' \in L[f,l]} (T^l + \frac{q^l}{B^l}), \tag{3.13}$$

where *L[f,l]* denotes the set of links between l-queue and the sender, passing through the receiver.To take this delay into account, we added two modes(slow-start delay and congestion-avoidance delay), in which the system remains between a drop occurs and it is detected.The timing variable t_{tim} is used to enforce that the system remains in the slow-start delay, congestion-avoidance delay and fast recovery modes for the required time.

The inputs to the TCP-Sack flow model atr the round trip time, the drop events and the corresponding drop detection delays and its outputs are the sending rates of the end-to-end flows.

3.2 Hybrid Modeling Framework for Queue Dynamics

Consider a communication network consisting of a set $\mathcal{N}$ of nodes connected by a set $\mathcal{L}$ of links. We consider all links as unidirectional and denote by $l := \vec{ij}$, the link from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$. Every link $l \in \mathcal{N}$ is characterized by a finite bandwidth B^l and a propagation delay T^l. We assume that the network is being loaded by a set F of end-to-end flows. Given a flow $f \in \mathcal{F}$ from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$, we denote by $r_f flow'ssendingrate$, i.e., the rate at which packets are generated and enter node i where the flow is initiated. Given a link $l \in \mathcal{L}$ in the path of the flow, we denote by r_f the rate at which packets from the flow go through the l-link. We call r_f the *l-link/f-flow* transmission rate. At each link, the sum of the link/flow transmission rates must not exceed the bandwidth B^l, i.e.,

$$\sum_{f \in \mathcal{F}} r_f^l \leq B^l, \forall l \in \mathcal{L}. \tag{3.14}$$

In general, teh flow sending rates r_f, $f \in \mathcal{F}$ are determined by congestion control mechanisms and the link/flow transmission rates r_f^l are determined by packet conservation laws to be derived shortly.Associated with each link $l \in \mathcal{L}$, there is a *queue* that temporarily holds packets before transmission. We denote by q_f^l the number of bytes in this queue that belong to the f-flow. The total number of bytes in the queue is then given by

$$q^l := \sum_{f \in \mathcal{F}} q_f^l, \forall l \in \mathcal{L}. \tag{3.15}$$

The queue can hold, at most, a finite number of bytes that we denote by q^l_{max}. When q^l reaches q^l_{max}, drops will occur.

For each flow $f \in \mathcal{F}$, we denote by RTT_f the *f-flowround-trip time* that elapses between a packet is sent and its acknowliedgement is received. The round trip time can also be determined by adding the propagation delays T^l and queuing times q^l/B^l of all links involved in one round trip. In particular,

$RTT_f = \sum_{l\in\mathcal{L}[f]}(T^l + \frac{q^l}{B^l}\Big)$,

where $\mathcal{L}[f]$ denotes teh set of links involved in one round trip for flow *f*.

Given a link $l \in \mathcal{L}$ in the path $\mathcal{L}[f]$, we denote by s^l_f teh rate at which flow packets arrive and call it the l-link/f-flow arrival rate. If l is the link where the flow starts then $s^l_f = r^l_f$, otherwise $s^l_f = r^{l'}_f$ with l' the previous link of l in the path of $\mathcal{L}[f]$.

3.2.1 Queue Dynamics

For the queue dynamics, we have the foolowing two assumptions.

ASSUMPTION 1 : The packets of different flows aarive at each node in their paths uniformly distributed over time.

ASSUMPTION 2 : Packets of different flows are uniformly distributed in each queue.

The continuous queue dynamics for a *l-link/f-flow* pair is given by

$\dot{q^l_f} = s^l_f - d^l_f - r^l_f$,

where d^l_f denotes the *f-flow drop rate.*

The discrete dynamics are given as follws with three discrete states.

Empty Queue ($q^l = 0$) : The outgoing rates r^l_f are equal to the arrival rates s^l_f and there are no drops as long as the bandwidth constrain (1.1) is not violated. When $\sum_{f\in\mathcal{F}} s^l_f \geq B^l$, the available bandwidth B^l is distributed among all flows proportionately to their arrival rates s^l_f by assumption 1. To summarise , we have

$$d^l_f = 0, r^l_f = \begin{cases} s^l_f & \sum_{f\in\mathcal{F}} s^l_f \leq B^l \\ \frac{s^l_f}{\sum_{\bar{f}\in\mathcal{F}} s^l_{\bar{f}}} B^l & \text{otherwise} \end{cases} \tag{3.16}$$

Queue neither empty nor full ($0 < q^l < q^l_{max}$): In this case the drops do not occur and because of assumption 2, the available link bandwidth B^l is distributed among the flows proportionally to their percentage of bytes in the queue.

$$d^l_f = 0, r^l_f = \frac{q^l_f}{\sum_{\bar{f}\in\mathcal{F}} q^l_{\bar{f}}} B^l. \tag{3.17}$$

Queue full and still filling ($q^l = q^l_{max}$ and $\sum_{f\in\mathcal{F}} s^l_f > B^l$): In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$d^l = \sum_{\bar{f}\in\mathcal{F}} s^l_{\bar{f}} - B^l > 0. \tag{3.18}$$

From assumption 1, the drop rate d^l should be distributed among all flows proportionally to their arrival rates s^l_f. Also from assumption 2, the available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarise, we have

$$d^l_f = \frac{s^l_f(\sum_{\bar{f}\in\mathcal{F}} s^l_{\bar{f}} - B^l)}{\sum_{\bar{f}\in\mathcal{F}} s^l_{\bar{f}}}, r^l_f = \frac{q^l_f}{\sum_{\bar{f}\in\mathcal{F}} q^l_{\bar{f}}} B^l. \tag{3.19}$$

To determine when and which flow suffers drops, suppose at time t_1, q^l reached q^l_{max} with $s^l := \sum_{\bar{f}\in\mathcal{F}} s^l_{\bar{f}} > B^l$. A drop will occur at time t_1, but, in general multiple drops may occur. In general, if a drop occured at time t_k a new drop is expected at time $t_{k+1} > t_k$ for which the total drop rate d^l integrates from t_k to t_{k+1} to the packet size L:

$$z^l := \int_{t_k}^{t_{k+1}} \sum_{f\in\mathcal{F}} d^l_f = \int_{t_k}^{t_{k+1}} (\sum_{f\in\mathcal{F}} s^l_f - B^l) = L. \tag{3.20}$$

To determine which flow suffers drop , we select a flow f_* where a drop occurs by drawing a fow randomly from the set $\mathcal{F}$ according to the distribution

$$p_{f_*}(f) = P(f_* = f) = \frac{d^l_f}{\sum_{\bar{f}\in\mathcal{F}} d^l_{\bar{f}}} = \frac{s^l_f}{\sum_{\bar{f}\in\mathcal{F}} s^l_{\bar{f}}}, \forall f \in \mathcal{F} \tag{3.21}$$

We assume that the flows $f_*(t_k)f_*(t_{k+1})$ that suffers drops at two distinct time instants t_k,t_{k+1} are conditionally independent random variables (given that the drops occured at

times t_k, t_{k+1}).

3.3 Hybrid Modeling Framework for Queue Dynamics in Priority Queuing

Consider a communication network consisting of just 2 nodes connected by a link in which QoS is implemented. It is assumed that the flows have already been classified and IP Precedence has been set. The congestion management mechanism used here is Priority Queuing.Congestion avoidance/Traffic Policing/Link efficiency mechanisms are not implemented in this model.
The hybrid model of Priority Queuing is just the extension of the Queue dynamics of a simple link described in Chapter 1.
In Priority Queuing the link consists of 4 software queues named High,Medium,Normal,Low for different flows based on their IP-Precedence setting.The dynamics for medium,normal and low queue are similar compared to the dynamics of high queue.
The assumptions made in chapter 1 for queue dynamics holds here also for each queue.
The continuous queue dynamics as mentioned in chapter 1 holds here also.

$$\dot{q^l_{i_f}} = s^l_{i_f} - d^l_{i_f} - r^l_{i_f} \text{ where } i \in h, m, n, lw$$

(h denotes high queue, m denotes medium queue, n denotes normal queue and lw denotes low queue) and

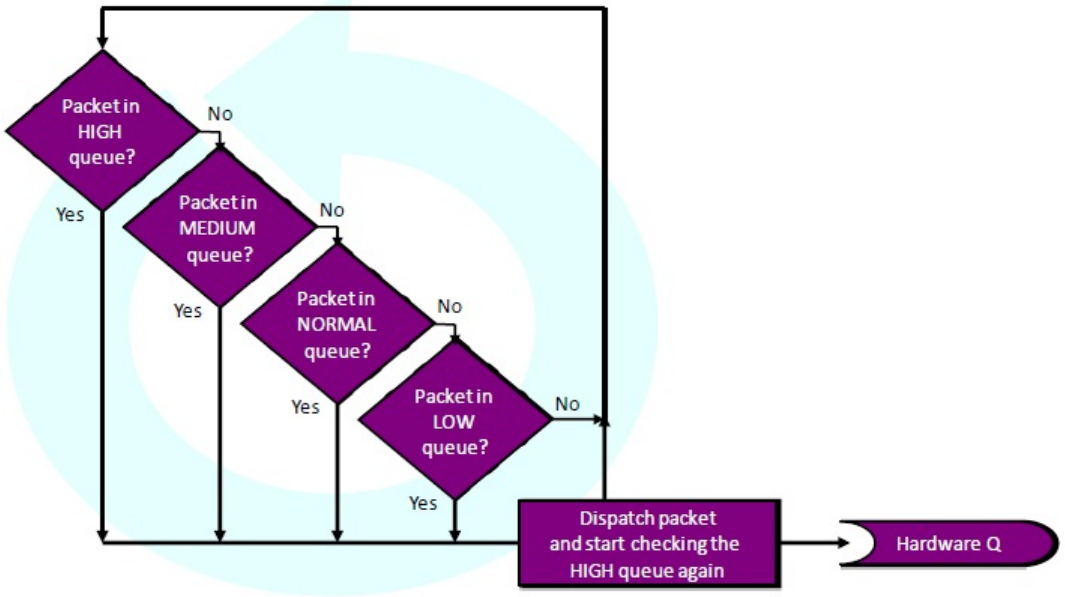
high queue denotes queue containing high priority flows,
medium queue denotes queue containing medium priority flows,
normal queue denotes queue containing normal priority flows,
low queue denotes queue containing low priority flows.

The following relations between the quantities are equivalent and these are used interchangeably in the forthcoming sections:

$$\sum_{f \in \mathcal{F}_i} s^l_{i_f} = s^l_i, \sum_{f \in \mathcal{F}_i} r^l_{i_f} = r^l_i, \sum_{f \in \mathcal{F}_i} q^l_{i_f} = q^l_i, \sum_{f \in \mathcal{F}_i} d^l_{i_f} = d^l_i$$

where $i \in h, m, n, lw$

FIGURE 3.2: Scheduling in Priority Queuing



3.3.1 High Queue Dynamics

As described in Chapter 2-Priority Queuing section,the High queue has the highest priority over all the other queues. The packet from the other queues will be transmitted only if there are no packets in the high queue. This is well explained in the figure 3.1. The different states of High Queue dynamics are as follows:

3.3.1.1 Queue Empty ($q_h^l = 0$)

The outgoing rates $r_{h_f}^l$ are equal to the arrival rates $s_{h_f}^l$ and there are no drops as long as the bandwidth constrain (1.1) is not violated. When $\sum_{f \in \mathcal{F}_h} s_{h_f}^l \geq B^l$, the available bandwidth B^l is distributed among all flows proportionately to their arrival rates $s_{h_f}^l$ by assumption 1. To summarise , we have

$$d_{h_f}^l = 0, r_{h_f}^l = \begin{cases} s_{h_f}^l & \sum_{f \in \mathcal{F}_h} s_{h_f}^l \leq B^l \\ \frac{s_{h_f}^l}{\sum_{f \in \mathcal{F}_h} s_{h_f}^l} B^l & \text{otherwise} \end{cases} \tag{3.22}$$

3.3.1.2 Queue neither empty nor full($0 < q_h^l < q_{h,max}^l$)

In this case the drops do not occur and because of assumption 2, the available link bandwidth B^l is distributed among the flows proportionally to their percentage of bytes in the queue.

$$d_{h_f}^l = 0, r_{h_f}^l = \frac{q_{h_f}^l}{\sum_{f \in \mathcal{F}_h} q_{h_f}^l} B^l. \tag{3.23}$$

3.3.1.3 Queue full and still filling($q_h^l = q_{h,max}^l$ and $\sum_{f \in \mathcal{F}_h} s_{h_f}^l > B^l$

In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$d_h^l = \sum_{f \in \mathcal{F}_h} s_{h_f}^l - B^l > 0. \tag{3.24}$$

From assumption 1, the drop rate d_h^l should be distributed among all flows proportionally to their arrival rates $s_{h_f}^l$. Also from assumption 2, the available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have

$$d_{h_f}^l = \frac{s_{h_f}^l (\sum_{f \in \mathcal{F}_h} s_{h_f}^l - B^l)}{\sum_{f \in \mathcal{F}_h} s_{h_f}^l}, r_{h_f}^l = \frac{q_{h_f}^l}{\sum_{f \in \mathcal{F}_h} q_{h_f}^l} B^l. \tag{3.25}$$

The hybrid model of high queue is given below.

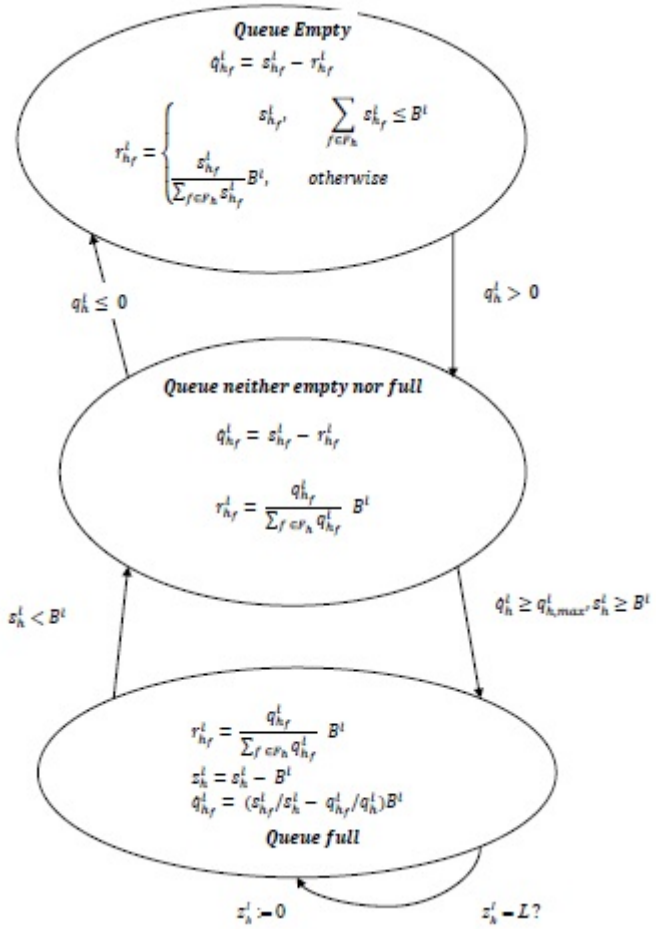
3.3.2 Dynamics of Medium Queue

The packets in the medium queue will be transmitted only if the high queue is empty.

3.3.2.1 Queue Empty ($q_m^l = 0$)

The outgoing rates $r_{m_f}^l$ are equal to the arrival rates $s_{m_f}^l$ and there are no drops as long as the bandwidth constrain (1.1) is not violated. When $\sum_{f \in \mathcal{F}_m} s_{m_f}^l + \sum_{f \in \mathcal{F}_h} s_{h_f}^l \geq B^l$ and $q_h^l \leq 0$, the available bandwidth B^l is distributed among all flows proportionately

FIGURE 3.3: Hybrid model of high queue dynamics



to their arrival rates $s^l_{m_f}$ by assumption 1. To summarise , we have

$$d^l_{m_f} = 0, r^l_{m_f} = \begin{cases} s^l_{m_f} & \sum_{f \in \mathcal{F}_m} s^l_{m_f} + \sum_{f \in \mathcal{F}_h} s^l_{h_f} \le B^l \\ \frac{s^l_{m_f}}{\sum_{f \in \mathcal{F}} s^l_{m_f}} (B^l - \sum_{f \in \mathcal{F}_h} r^l_{h_f}) & \text{otherwise and} q^l_h \le 0 \end{cases} \tag{3.26}$$

3.3.2.2 Queue neither empty nor full($0 < q^l_m < q^l_{m,max}$) and send($q^l_h \le 0$)

In this case the drops do not occur and because of assumption 2, the available link bandwidth B^l is distributed among the flows proportionally to their percentage of bytes in the queue.

$$d^l_{m_f} = 0, r^l_{m_f} = \frac{q^l_{m_f}}{\sum_{f \in \mathcal{F}} q^l_{m_f}} (B^l - \sum_{f \in \mathcal{F}_h} r^l_{h_f}). \tag{3.27}$$

3.3.2.3 Queue full and still filling($q_m^l = q_{m,max}^l$ and $\sum_{f\in\mathcal{F}_m} s_{m_f}^l > (B^l - \sum_{f\in\mathcal{F}_h} r_h^l)$ and $q_h^l \leq 0$

In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$z_m^l = \sum_{f\in\mathcal{F}_m} s_{m_f}^l - (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l) > 0. \tag{3.28}$$

From assumption 1, the drop rate d_m^l should be distributed among all flows proportionally to their arrival rates $s_{m_f}^l$. Also from assumption 2, the available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have

$$d_{m_f}^l = \frac{s_{m_f}^l(\sum_{f\in\mathcal{F}_m} s_{m_f}^l - (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l))}{\sum_{f\in\mathcal{F}_m} s_{m_f}^l}, r_{m_f}^l = \frac{q_{m_f}^l}{\sum_{f\in\mathcal{F}} q_{m_f}^l}(B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l). \tag{3.29}$$

3.3.2.4 Queue neither full nor empty($0 < q_m^l < q_{m,max}^l$) and not send($q_h^l > 0$)

Since the high queue is not empty($q_h^l > 0$), the flows in this medium queue will queue up and doesn't send.

$$d_{m_f}^l = 0, r_{m_f}^l = 0. \tag{3.30}$$

3.3.2.5 Queue full and still filling($q_m^l = q_{m,max}^l$ and $\sum_{f\in\mathcal{F}_m} s_{m_f}^l > (B^l - \sum_{f\in\mathcal{F}_h} r_h^l)$ and $q_h^l > 0$

When the high queue is not empty and the medium queue is full, the incoming flows are dropped.

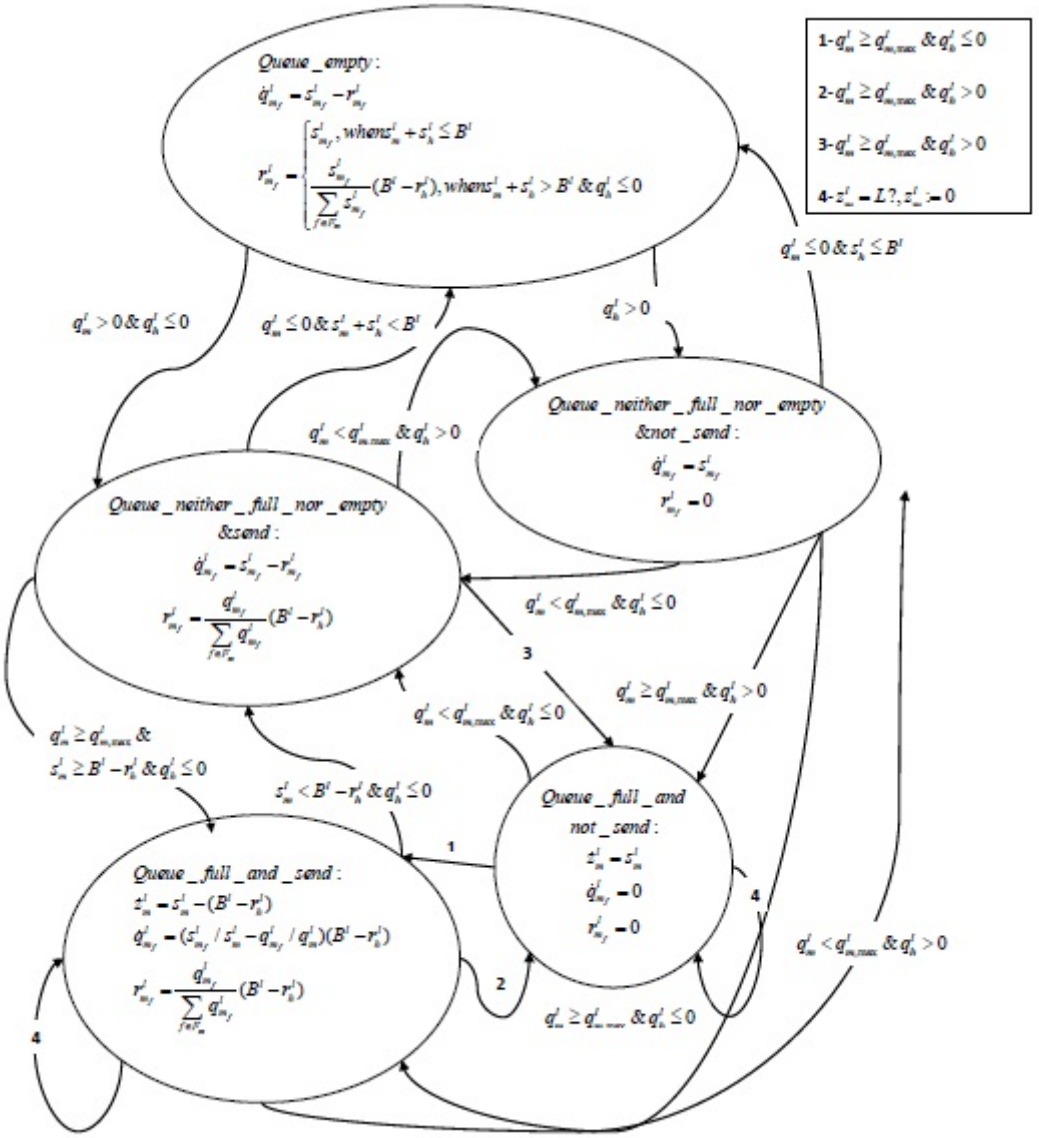
$$d_{m_f}^l = \frac{s_{m_f}^l(\sum_{f\in\mathcal{F}_m} s_{m_f}^l - (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l))}{\sum_{f\in\mathcal{F}_m} s_{m_f}^l}, r_{m_f}^l = 0. \tag{3.31}$$

The state-flow diagram for the medium queue is given below:

3.3.3 Dynamics of Normal Queue

The flows in the normal queue will be transmitted only if the high queue and medium queue are empty.

FIGURE 3.4: Hybrid model for the Queue containing medium priority flows



3.3.3.1 Queue Empty ($q^l_n = 0$)

The outgoing rates $r^l_{n_f}$ are equal to the arrival rates $s^l_{n_f}$ and there are no drops as long as the bandwidth constrain (1.1) is not violated. When $\sum_{f \in \mathcal{F}_n} s^l_{n_f} + \sum_{f \in \mathcal{F}_m} s^l_{m_f} + \sum_{f \in \mathcal{F}_h} s^l_{h_f} \ge B^l$ and $q^l_h \le 0$ and $q^l_m \le 0$, the available bandwidth B^l is distributed among all flows

proportionately to their arrival rates $s^l_{n_f}$ by assumption 1. To summarise , we have

$$
\begin{aligned}
d^l_{n_f} &= 0 \\
r^l_{n_f} &= s^l_{n_f} && \sum_{f\in\mathcal{F}_n} s^l_{n_f} + \sum_{f\in\mathcal{F}_m} s^l_{m_f} + \sum_{f\in\mathcal{F}_h} s^l_{h_f} \le B^l \\
&= \frac{s^l_{n_f}}{\sum_{f\in\mathcal{F}} s^l_{n_f}}(B^l - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_h} r^l_{h_f}) && \text{otherwise and} q^l_h \le 0, q^l_m \le 0
\end{aligned}
$$

3.3.3.2 Queue neither empty nor full($0 < q^l_n < q^l_{n,max}$) and send($q^l_h \le 0$ and $q^l_m \le 0$)

In this case the drops do not occur and because of assumption 2, the available link bandwidth B^l is distributed among the flows proportionally to their percentage of bytes in the queue.

$$d^l_{n_f} = 0, r^l_{n_f} = \frac{q^l_{n_f}}{\sum_{f\in\mathcal{F}} q^l_{n_f}}(B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f}). \tag{3.32}$$

3.3.3.3 Queue full and still filling($q^l_n = q^l_{n,max}$ and $\sum_{f\in\mathcal{F}_n} s^l_{n_f} > (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f})$ and $q^l_h \le 0$ and $q^l_m \le 0$

In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$z^l_n = \sum_{f\in\mathcal{F}_n} s^l_{n_f} - (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f}) > 0. \tag{3.33}$$

From assumption 1, the drop rate d^l_n should be distributed among all flows proportionally to their arrival rates $s^l_{n_f}$. Also from assumption 2, the available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have

$$
\begin{aligned}
d^l_{n_f} &= \frac{s^l_{n_f}(\sum_{f\in\mathcal{F}_n} s^l_{n_f} - (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f}))}{\sum_{f\in\mathcal{F}_n} s^l_{n_f}} \\
r^l_{n_f} &= \frac{q^l_{n_f}}{\sum_{f\in\mathcal{F}} q^l_{n_f}}(B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f}).
\end{aligned}
$$

3.3.3.4 Queue neither full nor empty($0 < q_n^l < q_{n,max}^l$) and not send($q_h^l > 0$ and $q_m^l > 0$)

Since the high queue and medium queue are not empty($q_h^l > 0$ and $q_m^l > 0$), the flows in this medium queue will queue up and doesn't send.

$$d_{n_f}^l = 0, r_{n_f}^l = 0. \tag{3.34}$$

3.3.3.5 Queue full and still filling($q_n^l = q_{n,max}^l$ and $\sum_{f\in\mathcal{F}_n} s_{n_f}^l > (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l)$ and $q_h^l > 0$ and $q_m^l > 0$

When the high queue is not empty and the medium queue is full, the incoming flows are dropped.

$$d_{n_f}^l = \frac{s_{n_f}^l(\sum_{f\in\mathcal{F}_n} s_{n_f}^l - (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l))}{\sum_{f\in\mathcal{F}_n} s_{n_f}^l}, r_{n_f}^l = 0. \tag{3.35}$$

The stateflow diagram of the queue containing normal priority flows will be the same as the one containing medium priority except that additional conditions as explained above will be added. Low Queue Dynamics

3.3.4 Dynamics of Low Queue

The flows in the normal queue will be transmitted only if the high,medium and normal queues are empty.

3.3.4.1 Queue Empty ($q_{lw}^l = 0$)

The outgoing rates $r_{lw_f}^l$ are equal to the arrival rates $s_{lw_f}^l$ and there are no drops as long as the bandwidth constrain (1.1) is not violated. When $\sum_{f\in\mathcal{F}_{lw}} s_{lw_f}^l + \sum_{f\in\mathcal{F}_n} s_{n_f}^l + \sum_{f\in\mathcal{F}_m} s_{m_f}^l + \sum_{f\in\mathcal{F}_h} s$ B^l and $q_h^l \leq 0$,$q_m^l \leq 0$ and $q_n^l \leq 0$, the available bandwidth B^l is distributed among all flows proportionately to their arrival rates $s_{lw_f}^l$ by assumption 1. To summarise , we

have

$$
\begin{aligned}
d^l_{lw_f} &= 0 \\
r^l_{lw_f} &= s^l_{lw_f} \text{when } \sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f} + \sum_{f\in\mathcal{F}_n} s^l_{n_f} + \sum_{f\in\mathcal{F}_m} s^l_{m_f} + \sum_{f\in\mathcal{F}_h} s^l_{h_f} \le B^l \\
&= \frac{s^l_{lw_f}}{\sum_{f\in\mathcal{F}} s^l_{lw_f}}(B^l - \sum_{f\in\mathcal{F}_n} r^l_{n_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_h} r^l_{h_f}), \text{otherwise and } q^l_h \le 0, q^l_m \le 0, q^l_n \le 0
\end{aligned}
$$

3.3.4.2 Queue neither empty nor full($0 < q^l_{lw} < q^l_{lw,max}$) and send($q^l_h \le 0$,$q^l_m \le 0$ and $q^l_n \le 0$)

In this case the drops do not occur and because of assumption 2, the available link bandwidth B^l is distributed among the flows proportionally to their percentage of bytes in the queue.

$$
d^l_{lw_f} = 0, r^l_{lw_f} = \frac{q^l_{lw_f}}{\sum_{f\in\mathcal{F}} q^l_{lw_f}}(B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_n} r^l_{n_f}). \tag{3.36}
$$

3.3.4.3 Queue full and still filling($q^l_{lw} = q^l_{lw,max}$ and $\sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f} > (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_n} r^l_{n_f})$ and $q^l_h \le 0$,$q^l_m \le 0$ and $q^l_n \le 0$

In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$
z^l_{lw} = \sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f} - (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_n} r^l_{n_f}) > 0. \tag{3.37}
$$

From assumption 1, the drop rate d^l_{lw} should be distributed among all flows proportionally to their arrival rates $s^l_{lw_f}$. Also from assumption 2, the available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have

$$
\begin{aligned}
d^l_{lw_f} &= \frac{s^l_{lw_f}(\sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f} - (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_n} r^l_{n_f}))}{\sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f}} \\
r^l_{lw_f} &= \frac{q^l_{lw_f}}{\sum_{f\in\mathcal{F}_{lw}} q^l_{lw_f}}(B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_n} r^l_{n_f}).
\end{aligned}
$$

3.3.4.4 Queue neither full nor empty($0 < q^l_{lw} < q^l_{lw,max}$) and not send($q^l_h > 0$,$q^l_m > 0$ and $q^l_n > 0$)

Since the high queue and medium queue are not empty($q^l_h > 0$,$q^l_m > 0$ and $q^l_n > 0$), the flows in this medium queue will queue up and doesn't send.

$$d^l_{lw_f} = 0, r^l_{lw_f} = 0. \tag{3.38}$$

3.3.4.5 Queue full and still filling($q^l_{lw} = q^l_{lw,max}$ and $\sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f} > (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_n} r^l_{n_f})$ and $q^l_h > 0$,$q^l_m > 0$,$q^l_n > 0$

When the high queue is not empty and the medium queue is full, the incoming flows are dropped.

$$d^l_{lw_f} = \frac{s^l_{lw_f}(\sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f} - (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f} - \sum_{f\in\mathcal{F}_m} r^l_{m_f} - \sum_{f\in\mathcal{F}_n} r^l_{n_f}))}{\sum_{f\in\mathcal{F}_{lw}} s^l_{lw_f}}, r^l_{lw_f} = 0. \tag{3.39}$$

The stateflow diagram of the queue containing low priority flows will be the same as the one containing medium priority except that additional conditions as explained above will be added.

3.4 Hybrid Modeling Framework for Queue Dynamics in Modified Deficit Round Robin

Consider a communication network consisting of just 2 nodes connected by a link in which QoS is implemented. It is assumed that the flows have already been classified and IP Precedence has been set. The congestion management mechanism used here is Modified Deficit Round Robin.Congestion avoidance/Traffic Policing/Link efficiency mechanisms are not implemented in this model.

The hybrid model of MDRR is just the extension of the Queue dynamics of priority queuing described in the previous section.

In MDRR the link consists of 3 software queues named Gold,Premium and Default for different flows based on their IP-Precedence setting.The dynamics for Premium and default queues are similar compared to the dynamics of Gold queue.

The assumptions made in chapter 1 for queue dynamics holds here also for each queue. The continuous queue dynamics as mentioned in chapter 1 holds here also.

$$\dot{q^l_{i_f}} = s^l_{i_f} - d^l_{i_f} - r^l_{i_f} \text{ where } i \in h, m, n, lw$$

(h denotes gold queue, m denotes Premium queue, n denotes default queue).

Gold queue denotes queue containing high priority flows(IP Precedence 5),
Premium queue denotes queue containing medium priority flows(IP Precedence 2,4,6 and 7),
Default queue denotes queue containing default priority flows(IP Precedence 1,3 and 0)

The following relations between the quantities are equivalent and these are used interchangeably in the forthcoming sections:

$$\sum_{f \in \mathcal{F}_i} s^l_{i_f} = s^l_i, \sum_{f \in \mathcal{F}_i} r^l_{i_f} = r^l_i, \sum_{f \in \mathcal{F}_i} q^l_{i_f} = q^l_i, \sum_{f \in \mathcal{F}_i} d^l_{i_f} = d^l_i$$

where $i \in h, m, n$

3.4.1 Gold Queue Dynamics

The hybrid model of high queue is same as described in section 3.3.1.

3.4.2 Dynamics of Premium Queue

The packets in the Premium queue will be transmitted only if the high queue is empty.

3.4.2.1 Queue Empty ($q^l_m = 0$)

The outgoing rates $r^l_{m_f}$ are equal to the arrival rates $s^l_{m_f}$ and there are no drops as long as the bandwidth constrain (1.1) is not violated. When $\sum_{f \in \mathcal{F}_m} s^l_{m_f} \geq (B^l - \sum_{f \in \mathcal{F}_h} r^l_{h_f}) * 0.8$ and $q^l_h \leq 0$, 80% of the available bandwidth is distributed among all flows proportionately to their arrival rates $s^l_{m_f}$ by assumption 1. To summarise , we

have

$$d^l_{m_f} = 0, r^l_{m_f} = \begin{cases} s^l_{m_f} & \sum_{f\in\mathcal{F}_n} s^l_{n_f} + \sum_{f\in\mathcal{F}_m} s^l_{m_f} + \sum_{f\in\mathcal{F}_h} s^l_{h_f} \leq B^l \\ \frac{s^l_{m_f}}{\sum_{f\in\mathcal{F}} s^l_{m_f}}(B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f}) * 0.8 & \sum_{f\in\mathcal{F}_m} s^l_{m_f} \geq (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f}) * 0.8 \text{and} q^l_h \leq 0 \end{cases} \tag{3.40}$$

3.4.2.2 Queue neither empty nor full($0 < q^l_m < q^l_{m,max}$) and send($q^l_h \leq 0$)

In this case the drops do not occur and because of assumption 2, 80% of available bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue.

$$d^l_{m_f} = 0, r^l_{m_f} = \frac{q^l_{m_f}}{\sum_{f\in\mathcal{F}} q^l_{m_f}}(B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f}) * 0.8. \tag{3.41}$$

3.4.2.3 Queue full and still filling and send($q^l_m = q^l_{m,max}$ and $\sum_{f\in\mathcal{F}_m} s^l_{m_f} > 0.8 * (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f})$ and $q^l_h \leq 0$

In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$z^l_m = \sum_{f\in\mathcal{F}_m} s^l_{m_f} - 0.8 * (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f}) > 0. \tag{3.42}$$

From assumption 1, the drop rate d^l_m should be distributed among all flows proportionally to their arrival rates $s^l_{m_f}$. Also from assumption 2, the percentage of available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have

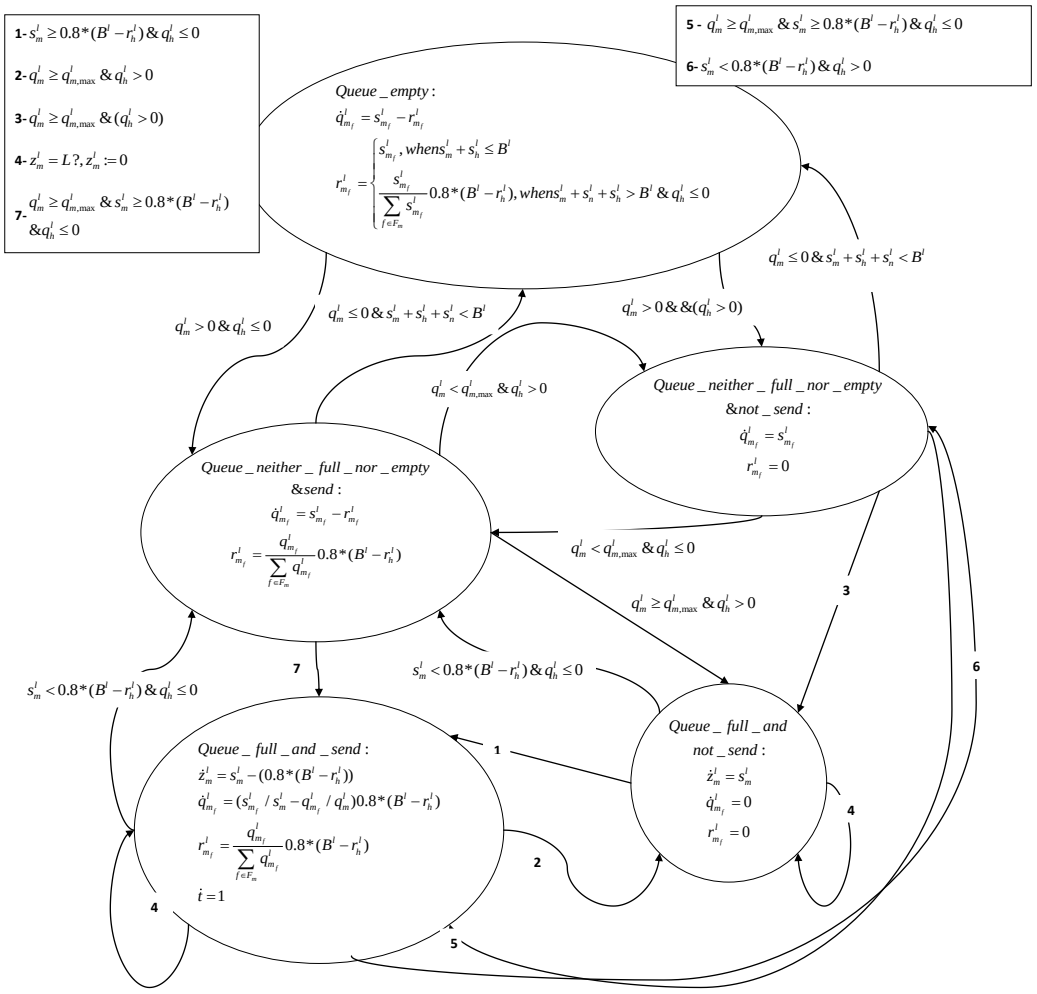
$$d^l_{m_f} = \frac{s^l_{m_f}(\sum_{f\in\mathcal{F}_m} s^l_{m_f} - 0.8 * (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f}))}{\sum_{f\in\mathcal{F}_m} s^l_{m_f}}, r^l_{m_f} = \frac{q^l_{m_f}}{\sum_{f\in\mathcal{F}} q^l_{m_f}} 0.8 * (B^l - \sum_{f\in\mathcal{F}_h} r^l_{h_f}). \tag{3.43}$$

3.4.2.4 Queue neither full nor empty($0 < q^l_m < q^l_{m,max}$) and not send($q^l_h > 0$)

Since the high queue is not empty($q^l_h > 0$), the flows in this Premium queue will queue up and doesn't send.

$$d^l_{m_f} = 0, r^l_{m_f} = 0. \tag{3.44}$$

FIGURE 3.5: Hybrid model for the Queue containing Premium priority flows



3.4.2.5 Queue full and still filling and not send($q_m^l = q_{m,max}^l$ and $\sum_{f \in \mathcal{F}_m} s_{m_f}^l > 0.8 * (B^l - \sum_{f \in \mathcal{F}_h} r_{h_f}^l)$ and $q_h^l > 0$

When the high queue is not empty and the Premium queue is full, the incoming flows are dropped.

$$d_{m_f}^l = \frac{s_{m_f}^l (\sum_{f \in \mathcal{F}_m} s_{m_f}^l - 0.8 * (B^l - \sum_{f \in \mathcal{F}_h} r_{h_f}^l))}{\sum_{f \in \mathcal{F}_m} s_{m_f}^l}, r_{m_f}^l = 0. \tag{3.45}$$

The state-flow diagram for the Premium queue is given below:

3.4.3 Dynamics of Default Queue

The flows in the default queue will be transmitted only if the high queue is empty.

3.4.3.1 Queue Empty ($q_n^l = 0$)

The outgoing rates $r_{n_f}^l$ are equal to the arrival rates $s_{n_f}^l$ and there are no drops as long as the bandwidth constrain (1.1) is not violated. When $\sum_{f\in\mathcal{F}_n} s_{n_f}^l \geq 0.2*(B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l)$ and $q_h^l \leq 0$ and $q_m^l > 0$, 20% of the available bandwidth is distributed among all flows proportionately to their arrival rates $s_{n_f}^l$ by assumption 1. To summarise , we have

$$
\begin{aligned}
&d_{n_f}^l = 0 \\
&r_{n_f}^l = s_{n_f}^l && \sum_{f\in\mathcal{F}_n} s_{n_f}^l + \sum_{f\in\mathcal{F}_m} s_{m_f}^l + \sum_{f\in\mathcal{F}_h} s_{h_f}^l \leq B^l \\
&\quad = \frac{s_{n_f}^l}{\sum_{f\in\mathcal{F}} s_{n_f}^l} 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l) && \sum_{f\in\mathcal{F}_n} s_{n_f}^l \geq 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l) \text{and} q_h^l \leq 0 q_m^l > 0 \\
&\quad = \frac{s_{n_f}^l}{\sum_{f\in\mathcal{F}} s_{n_f}^l} (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l) && \sum_{f\in\mathcal{F}_n} s_{n_f}^l \geq (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l) q_h^l \leq 0 q_m^l \leq 0
\end{aligned}
$$

3.4.3.2 Queue neither empty nor full($0 < q_n^l < q_{n,max}^l$) and send($q_h^l \leq 0$) and $q_m^l > 0$

In this case the drops do not occur and because of assumption 2, 20% of available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue.

$$d_{n_f}^l = 0, r_{n_f}^l = \frac{q_{n_f}^l}{\sum_{f\in\mathcal{F}} q_{n_f}^l} 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l). \tag{3.46}$$

3.4.3.3 Queue neither empty nor full($0 < q_n^l < q_{n,max}^l$) and send($q_h^l \leq 0$)and $q_m^l \leq 0$

In this case the drops do not occur and because of assumption 2, available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue.

$$d_{n_f}^l = 0, r_{n_f}^l = \frac{q_{n_f}^l}{\sum_{f\in\mathcal{F}} q_{n_f}^l} (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l). \tag{3.47}$$

3.4.3.4 Queue full and still filling($q_n^l = q_{n,max}^l$ and $\sum_{f\in\mathcal{F}_n} s_{n_f}^l > 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l$ and $q_h^l \leq 0$ and $q_m^l > 0$

In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$z_n^l = \sum_{f\in\mathcal{F}_n} s_{n_f}^l - (0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l)) > 0. \tag{3.48}$$

From assumption 1, the drop rate d_n^l should be distributed among all flows proportionally to their arrival rates $s_{n_f}^l$. Also from assumption 2, 20% of available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have

$$d_{n_f}^l = \frac{s_{n_f}^l(\sum_{f\in\mathcal{F}_n} s_{n_f}^l - 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l))}{\sum_{f\in\mathcal{F}_n} s_{n_f}^l}$$

$$r_{n_f}^l = \frac{q_{n_f}^l}{\sum_{f\in\mathcal{F}} q_{n_f}^l} 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l).$$

3.4.3.5 Queue full and still filling($q_n^l = q_{n,max}^l$ and $\sum_{f\in\mathcal{F}_n} s_{n_f}^l > (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l$ and $q_h^l) \leq 0$ and $q_m^l \leq 0$)

In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.

$$z_n^l = \sum_{f\in\mathcal{F}_n} s_{n_f}^l - (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l) > 0. \tag{3.49}$$

From assumption 1, the drop rate d_n^l should be distributed among all flows proportionally to their arrival rates $s_{n_f}^l$. Also from assumption 2,available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have

$$d_{n_f}^l = \frac{s_{n_f}^l(\sum_{f\in\mathcal{F}_n} s_{n_f}^l - (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l))}{\sum_{f\in\mathcal{F}_n} s_{n_f}^l}$$

$$r_{n_f}^l = \frac{q_{n_f}^l}{\sum_{f\in\mathcal{F}} q_{n_f}^l}(B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l - \sum_{f\in\mathcal{F}_m} r_{m_f}^l).$$

3.4.3.6 Queue neither full nor empty($0 < q_n^l < q_{n,max}^l$) and not send($q_h^l > 0$)

Since the high queue is not empty($q_h^l > 0$), the flows in this default queue will queue up and doesn't send.

$$d_{n_f}^l = 0, r_{n_f}^l = 0. \tag{3.50}$$

3.4.3.7 Queue full and still filling($q_n^l = q_{n,max}^l$ and $\sum_{f\in\mathcal{F}_n} s_{n_f}^l > 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l)$ and $q_h^l > 0$

When the high queue is not empty and the default queue is full, the incoming flows are dropped.

$$d_{n_f}^l = \frac{s_{n_f}^l(\sum_{f\in\mathcal{F}_n} s_{n_f}^l - 0.2 * (B^l - \sum_{f\in\mathcal{F}_h} r_{h_f}^l))}{\sum_{f\in\mathcal{F}_n} s_{n_f}^l}, r_{n_f}^l = 0. \tag{3.51}$$

The state-flow diagram for the default queue is given below:

3.5 Full network Model

In this chapter, we developed hybrid models for network traffic flows, queues and TCP packet sources.By composing them, the entire network model can be constructed.
In our model, input data is from Telecom Italia. In addition to it, a TCP flow of video streaming service [12] is also given as an input.The figure illustrates the full network model with the inputs.

FIGURE 3.6: Hybrid model for the Queue containing Default priority flows

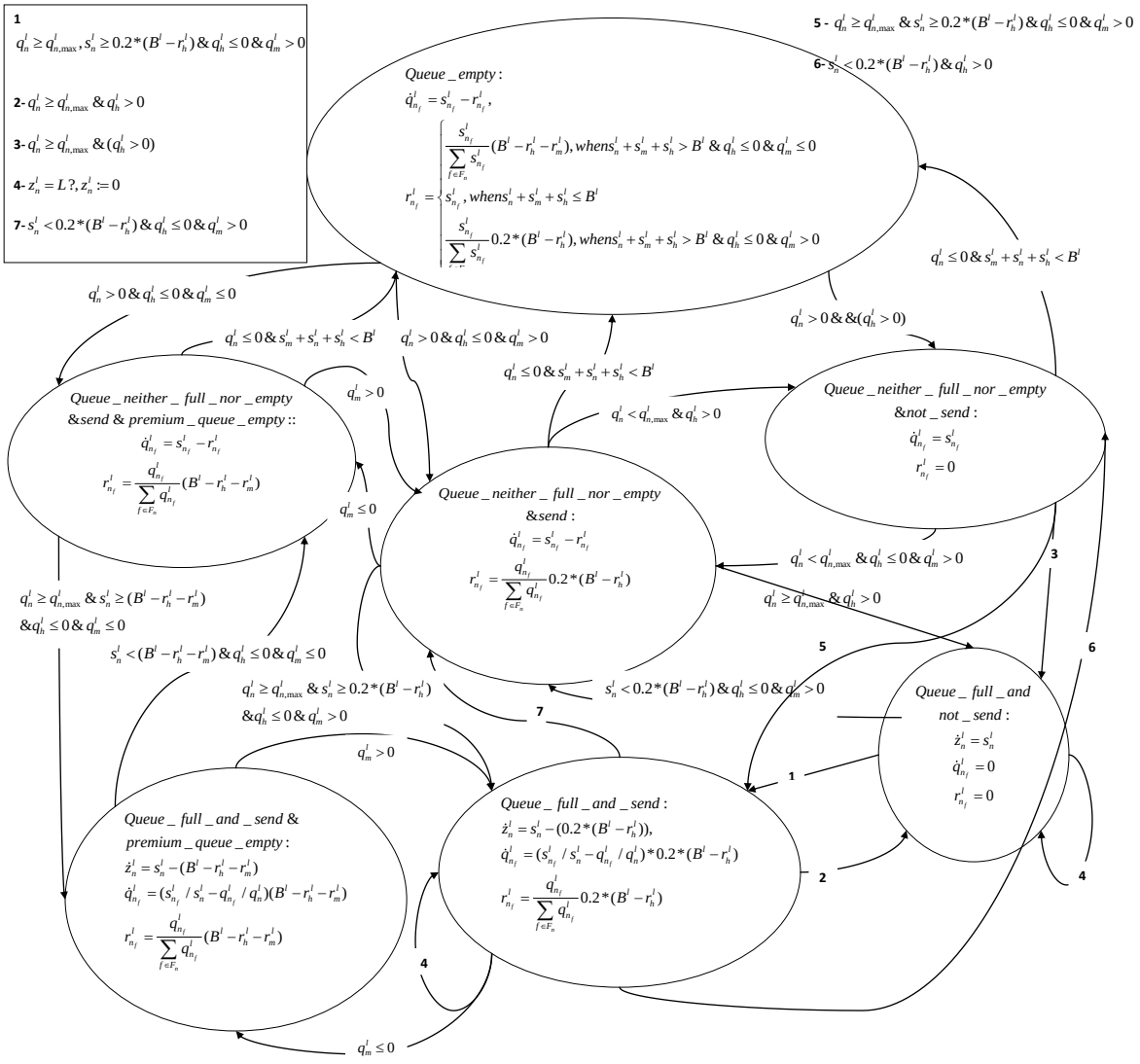
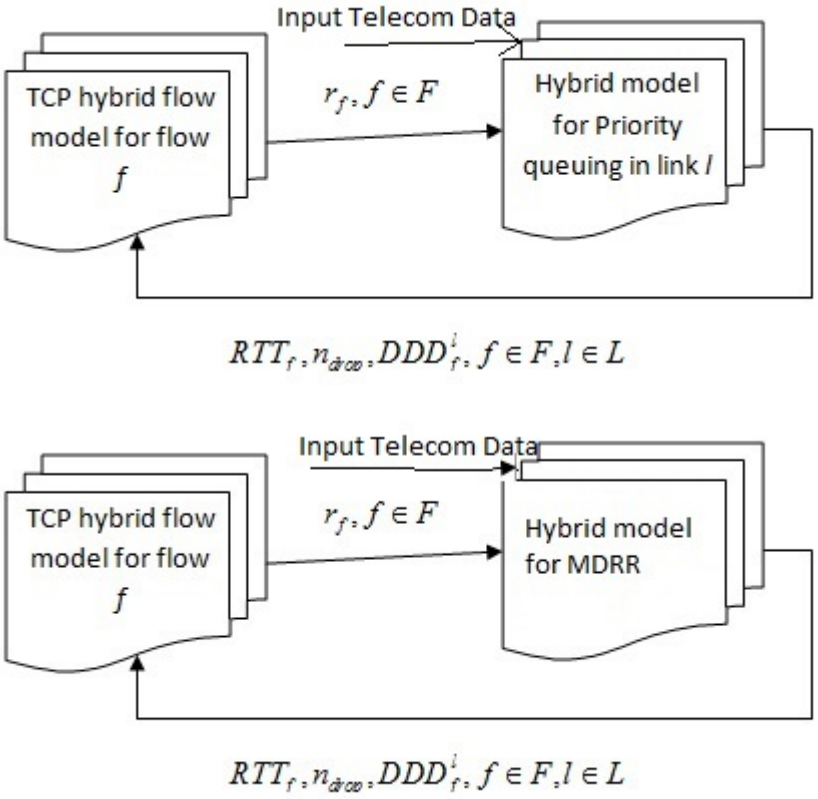


FIGURE 3.7: Overall Hybrid Model for TCP flow in Priority Queuing and MDRR based network



Chapter 4

Simulation of the Hybrid Model using MATLAB-SimuLink

4.1 Implementation of Hybrid Model

MATLAB-Simulink is used to implement this hybrid modeling framework of Priority queuing and MDRR. Since the MDRR consists of three queues and to make the comparison more meaningful, priority queuing model is also developed with three queues.This Hybrid model is developed for a *Telecom Italia router* in which the QoS is implemented.The router has input flows from four different links and its output is split and given to two other links.These inputs from 4 different links are summed up.Each input link has a bandwidth of 10Gbps. Since all these input links are summed up in our model,the total input link bandwidth is 40Gbps.

The input to this model are the real time data from Telecom Italia for flows with IP-Precedence 0 to 7 and a TCP flow for video streaming service was added for perturbation and we will be measuring QoE for such video streaming flows.The real time data from Telecom Italia was recorded for 2 days with interval time of 5 mins. The TCP video streaming service is given a priority of 4 [1] and is assigned to the medium queue.

The Simulink model consists of three stateflow charts, each for high/Gold,medium/premium and normal/Default queues.All the stateflow charts run in parallel and the input to each stateflow chart are their respective flow arrival rates s_f^l and the transport delay and the outputs are q_f^l,r_f^l.

Flow with IP Precedence 5 is given as input to the High/Gold Queue,flows with IP Precedence 2,4,6,7 are given as input to the medium/Premium queue and flows with IP Precedence 1,3,0 are given as input to the normal/Default queue.
When both the input and output link bandwidths are the same, there is no bottleneck and analysis of flow behavior makes no sense. So, we simulated to create a bottleneck scenario, where one of the output link fails.Therefore all the input flows has to be sent through this output link.
The Bandwidth of the output link used in this model: 0.5Gbps or 8.33Mbpm
Simulation time : 2880 mins (almost 2 days)
TCP flow created with maximum rate of 1Mbpm and it lasts for 15 mins and this is repeated for every hour.

4.1.1 MATLAB Model of MDRR

The overall MATLAB model looks like as depicted in the figure 4.1.

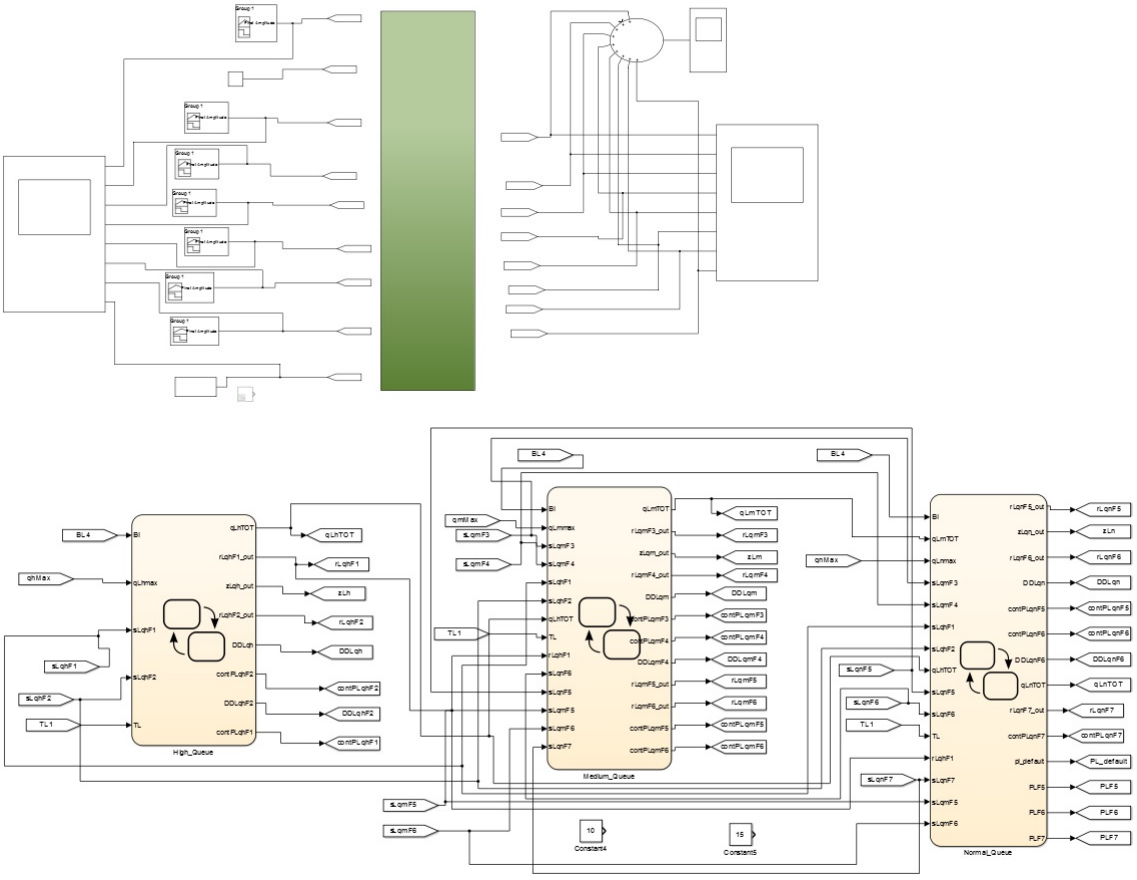
4.1.1.1 Behavior of Different flows(IP-Precedence 0-7)

The figures 4.2-4.8 depict flow behavior in the Premium queue and default queue. Due to the bottleneck, flows in the premium and default queue suffer drops. In this case, the total bandwidth is used when the Premium queue and the gold queue are not empty.The figures 4.9,4.10 depicts the queue sizes and the total bandwidth utilised.

4.1.1.2 Behavior of TCP flow for Video Streaming Service

Since the Premium priority flows suffer drops, the TCP flow for video streaming service also suffers drops. The figure 4.12 shows the queuing delay of the TCP flow. Since there are some packet losses,we should expect that the QoE decrease. Lets see that from the results of subjective and objective measurement techniques.
Using the subjective measurement estimation of QoE, we have the results in the figure 4.13. When there are drops, the PVQ decreases.
Also from the objective measurement technique, we have the results in the figures 4.14,4.15.We can see that the PSNR decreases as we have more packet loss rate and when the video bit rate decreases. The PVQ rating based on the two different PSNR's

FIGURE 4.1: MDRR MATLAB Model



are given in the figures 4.16,4.17.

4.1.2 MATLAB model of Priority Queuing

The overall MATLAB model looks like as depicted in the figure 4.18.

4.1.2.1 Behavior of Different flows(IP-Precedence 0-7)

Due to the bottleneck, there are drops in medium queue and high queue as well.However drops in high queue are significantly less compared to losses in medium and normal queue.The flow behaviors are depicted in the figures 4.19-4.26.

FIGURE 4.2: MDRR:IP Precedence 2 - Flow behavior

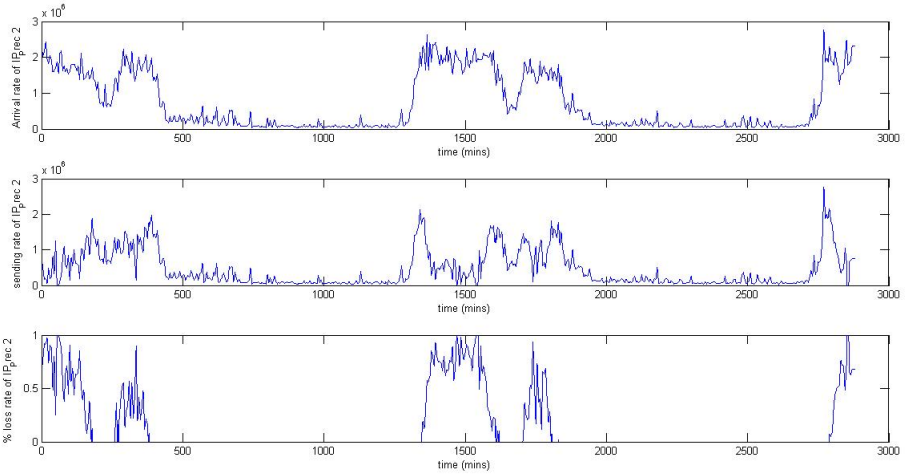


FIGURE 4.3: MDRR:IP Precedence 4 - Flow behavior

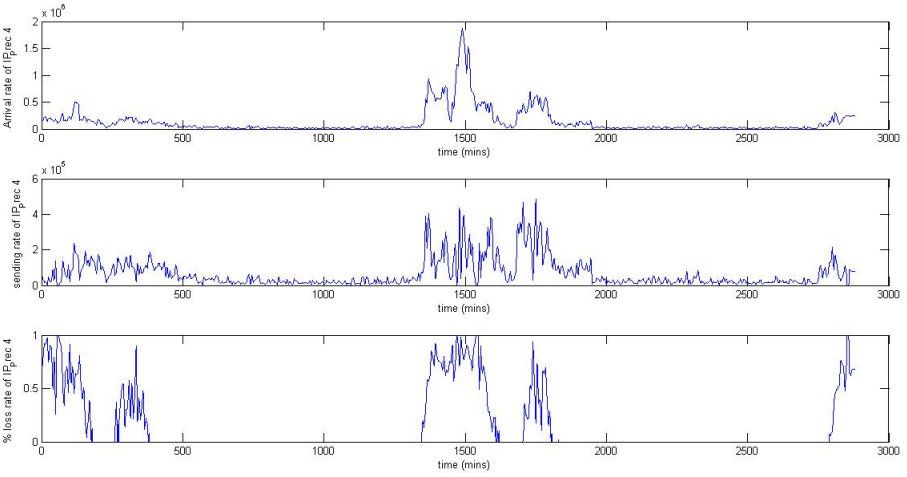


FIGURE 4.4: MDRR:IP Precedence 6 - Flow behavior

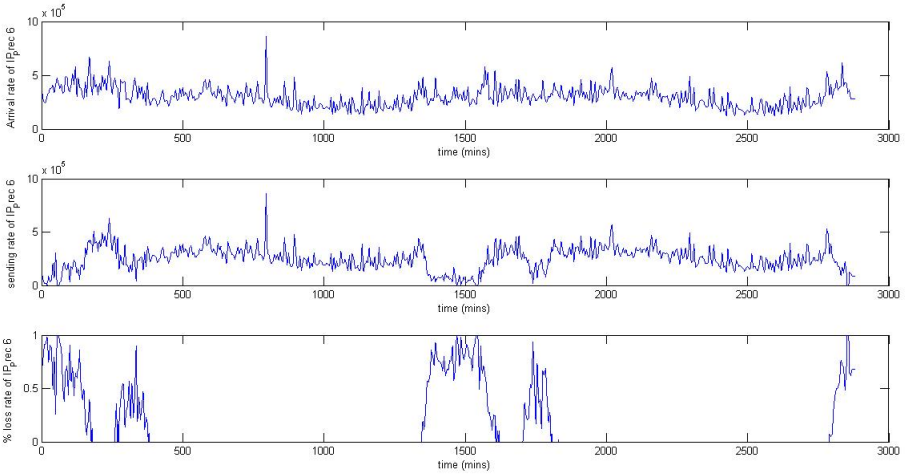
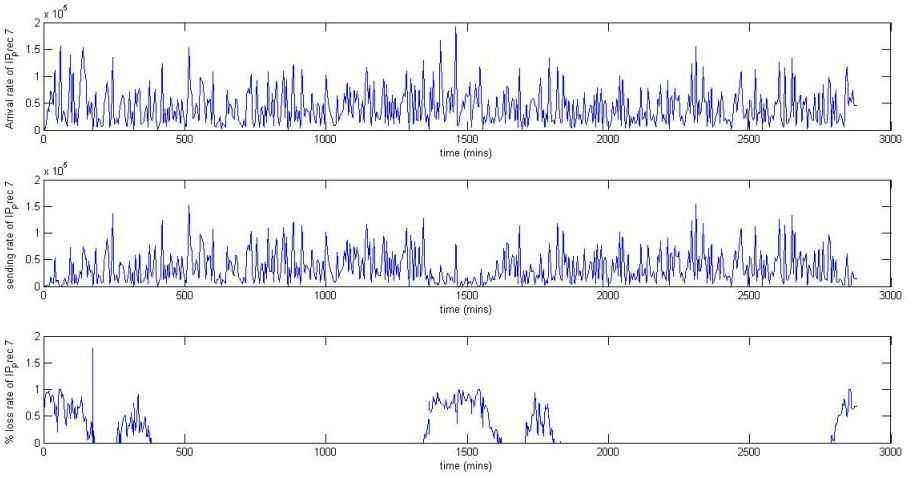


FIGURE 4.5: MDRR:IP Precedence 7 - Flow behavior



Since the normal queue is filling and full, the available bandwidth is utilised by it.

4.1.2.2 Scenario 2 PQ Results: Behavior of TCP flow for Video Streaming Service

As the medium queue is suffering drops,as shown in the figure 4.27, the TCP flow for Video streaming services also suffers drops.The flow behavior is shown in the figure 4.28. The figure 4.29 shows the queuing delay of the TCP flow. Let us examine the QoE with this flow behavior of TCP.

FIGURE 4.6: MDRR:IP Precedence 1 - Flow behavior

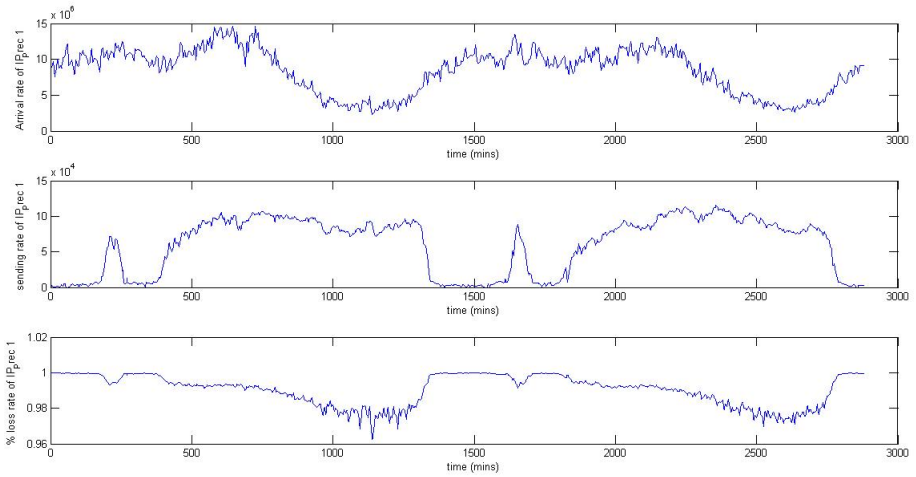


FIGURE 4.7: MDRR:IP Precedence 3 - Flow behavior

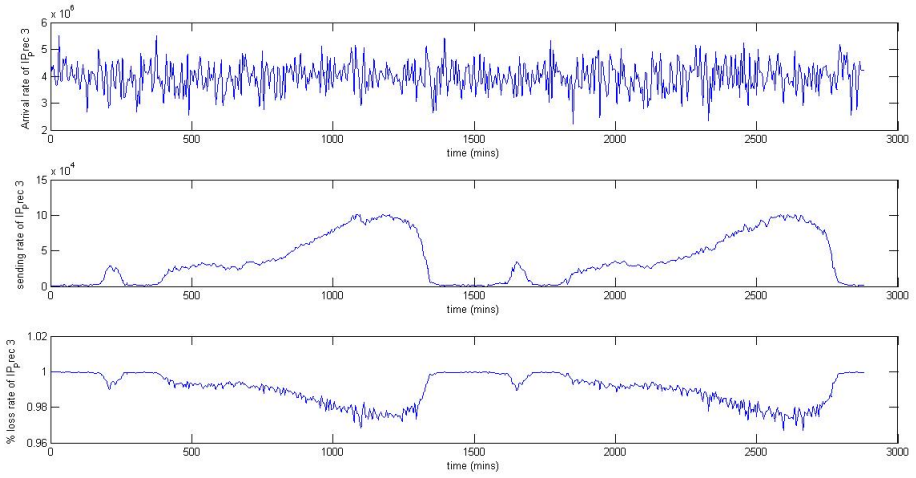


FIGURE 4.8: MDRR:IP Precedence 0 - Flow behavior

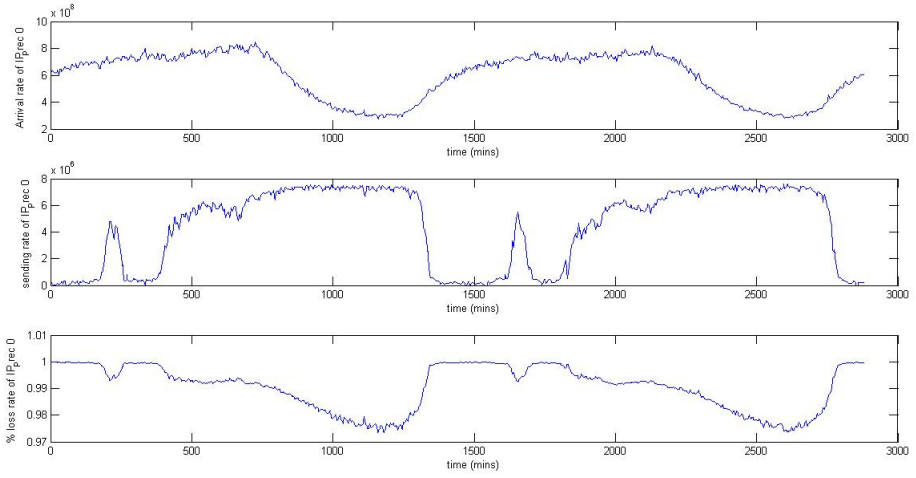


FIGURE 4.9: MDRR:Queue sizes

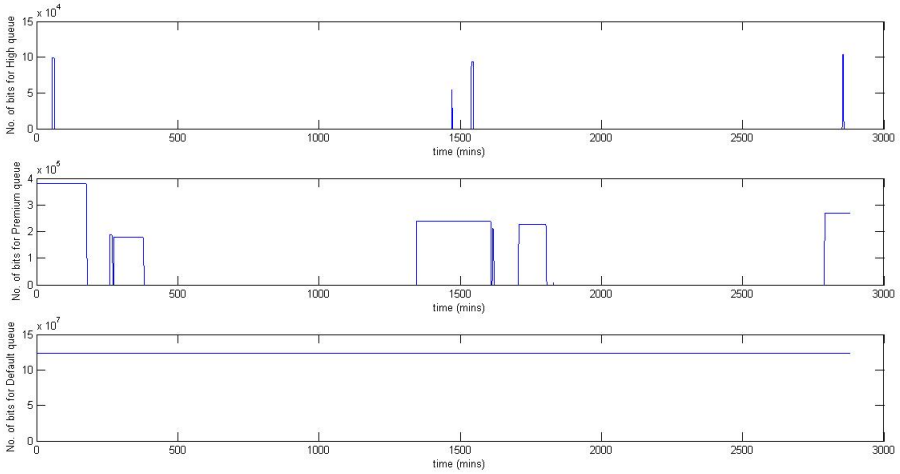


FIGURE 4.10: MDRR:Bandwidth Allocation

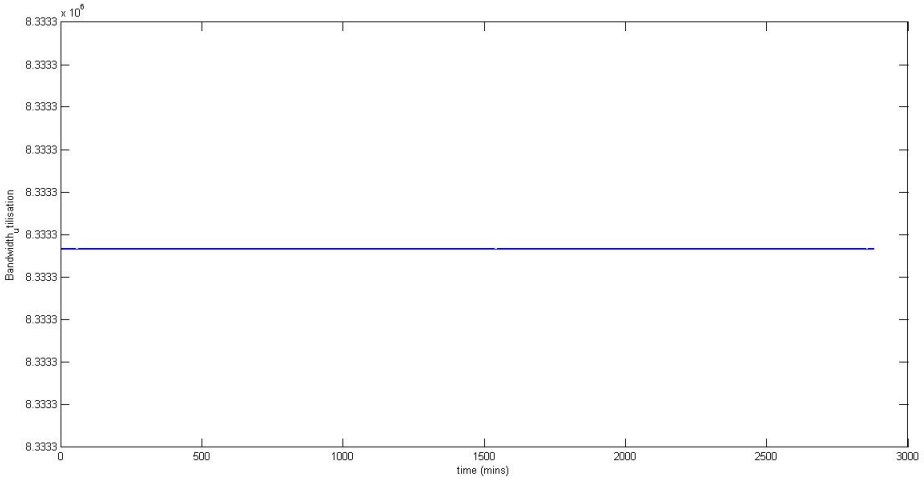


FIGURE 4.11: MDRR:TCP Flow behavior for Video Streaming Service

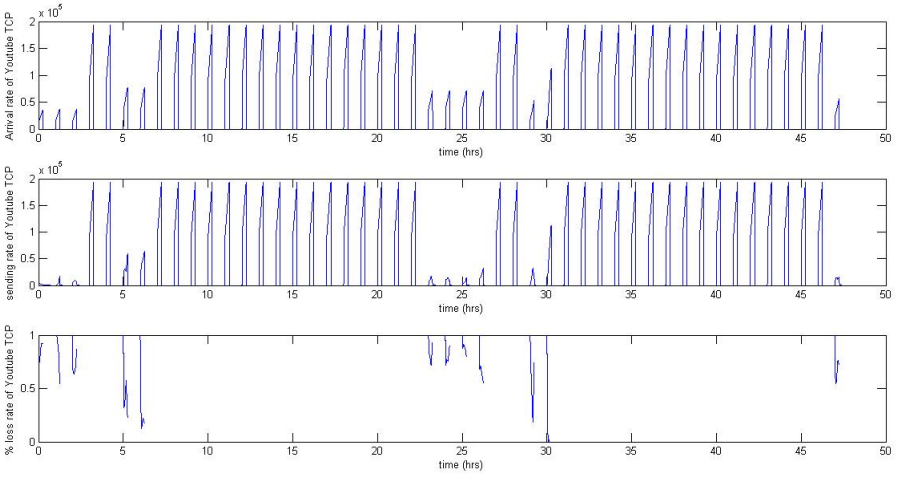


FIGURE 4.12: MDRR:Queuing delay of TCP flow for Video Streaming Service

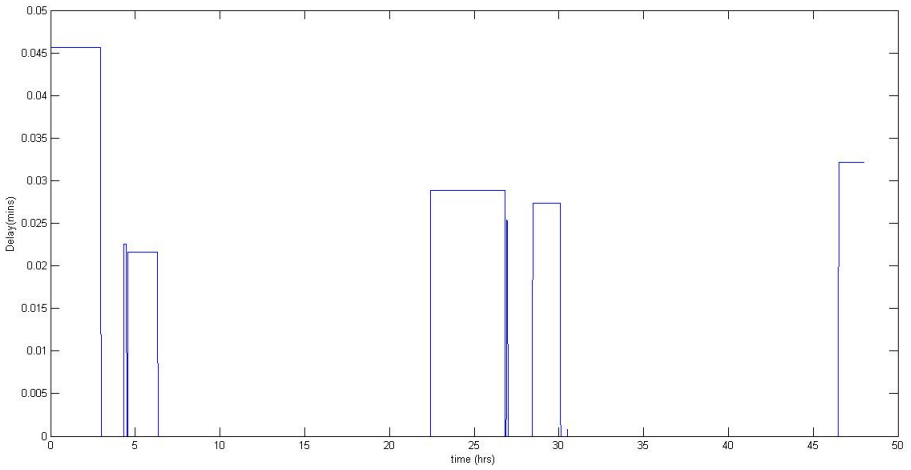
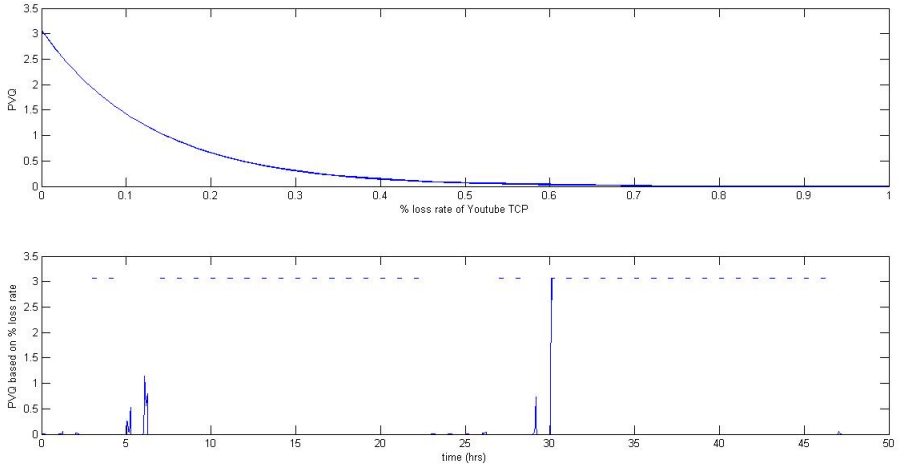


FIGURE 4.13: MDRR:PVQ and Packet Loss rate %



QoE rating decreases(same as MDRR) since there are packet losses and less bandwidth available.

As we have analysed two different queuing methods for two scenarios, it is now time to compare them.

4.1.3 Comparison of Priority Queuing and MDRR

The comparison is based on the following parameters.

Packet loss rate %

QoE of Video Streaming services

Queuing Delay

FIGURE 4.14: MDRR:PSNR and Packet Loss rate %

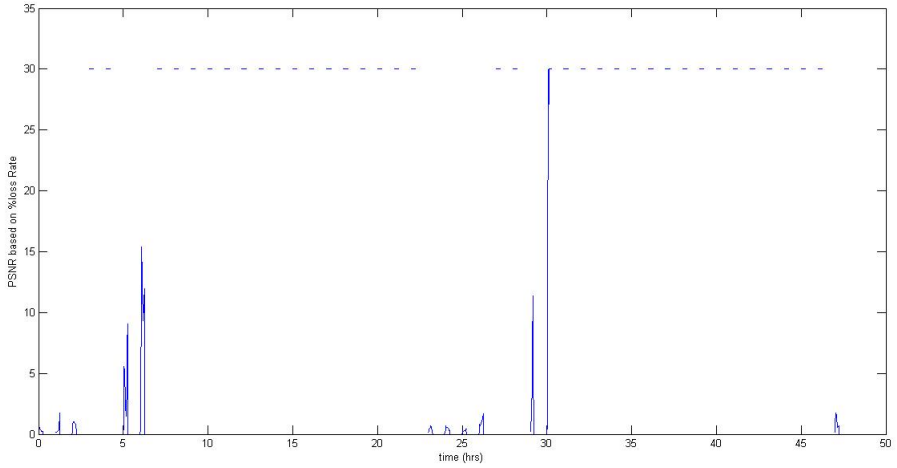
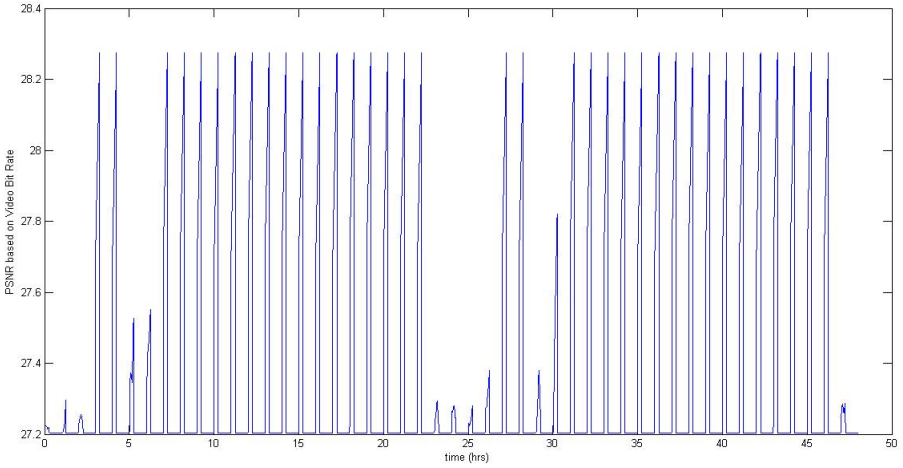


FIGURE 4.15: MDRR:PSNR and Video Bit Rate



- **Packet loss rate %** - There are more packet losses in priority queuing compared to MDRR.
- **QoE of Video Streaming services and Queuing Delay** - When the results were closely observed QoE fluctuates more frequently in priority queuing due to the fluctuation of queuing delay.Therefore, QoE is poor with priority queuing.Thus the user might experience delays when the video is loading.

FIGURE 4.16: MDRR:PVQ and PSNR(Packet Loss rate %)

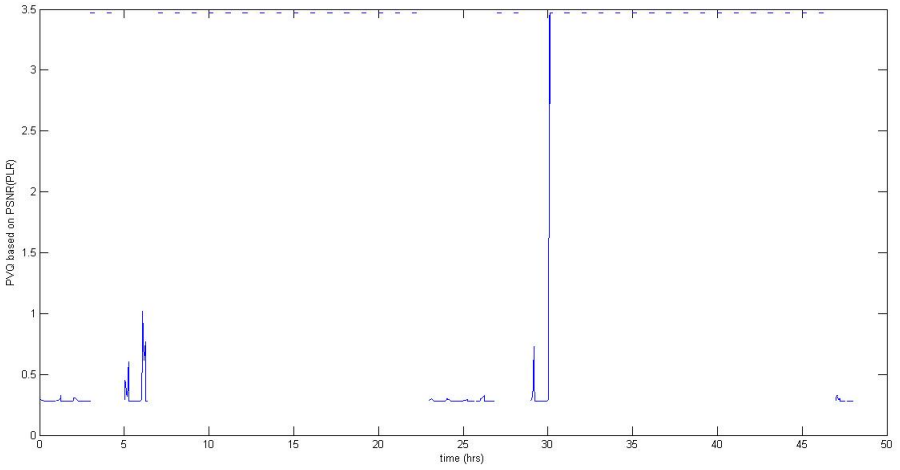


FIGURE 4.17: MDRR:PVQ and PSNR(Video Bit Rate)

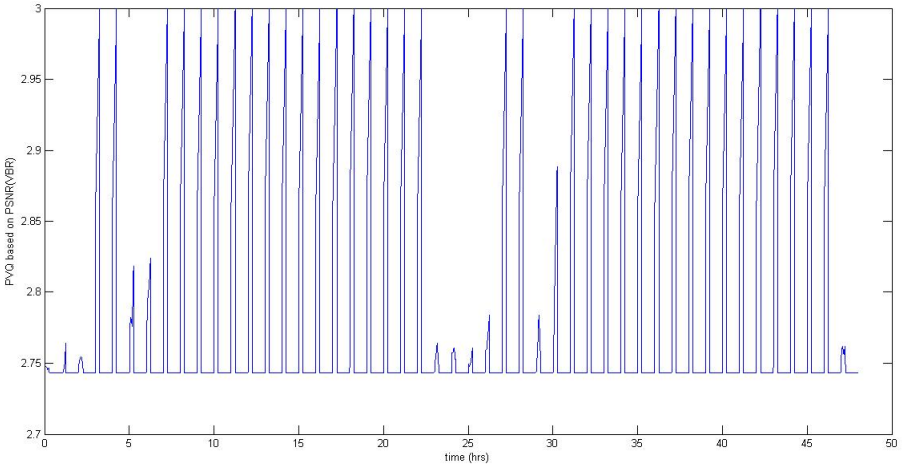


FIGURE 4.18: Priority Queuing MATLAB Model

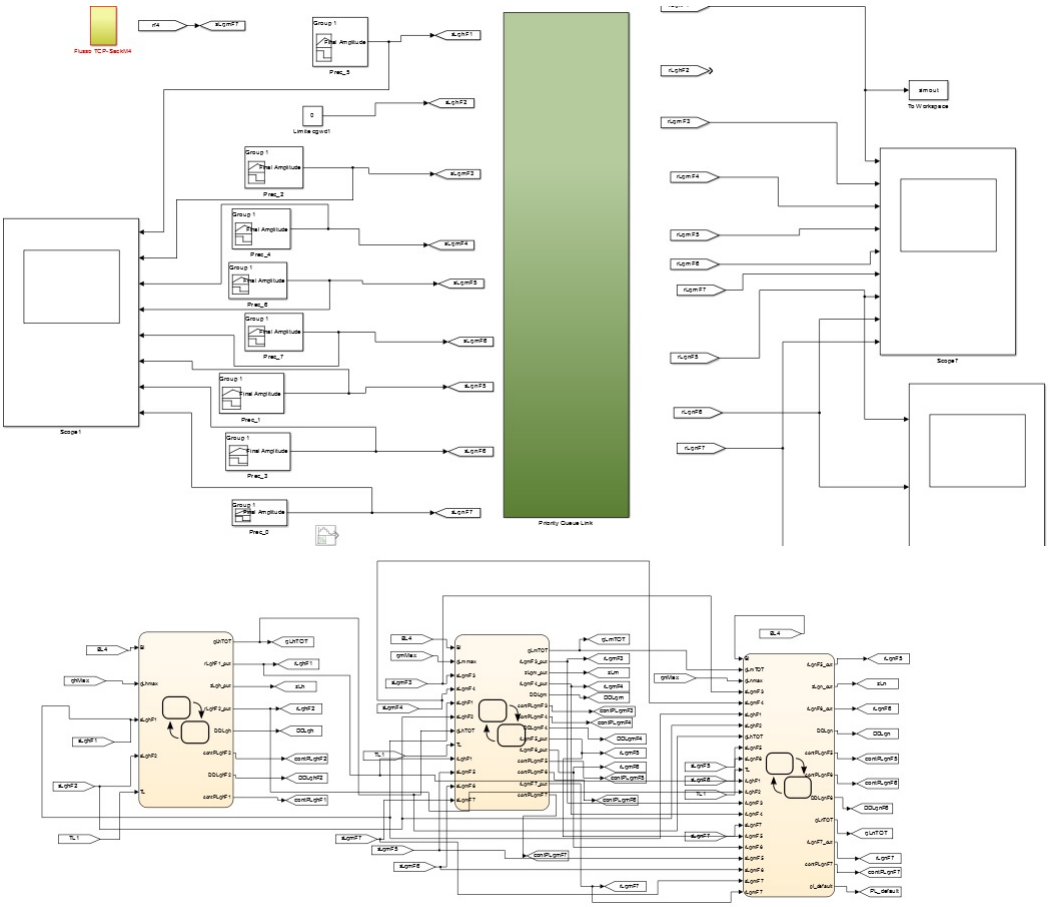


FIGURE 4.19: PQ:IP Precedence 0 - Flow behavior

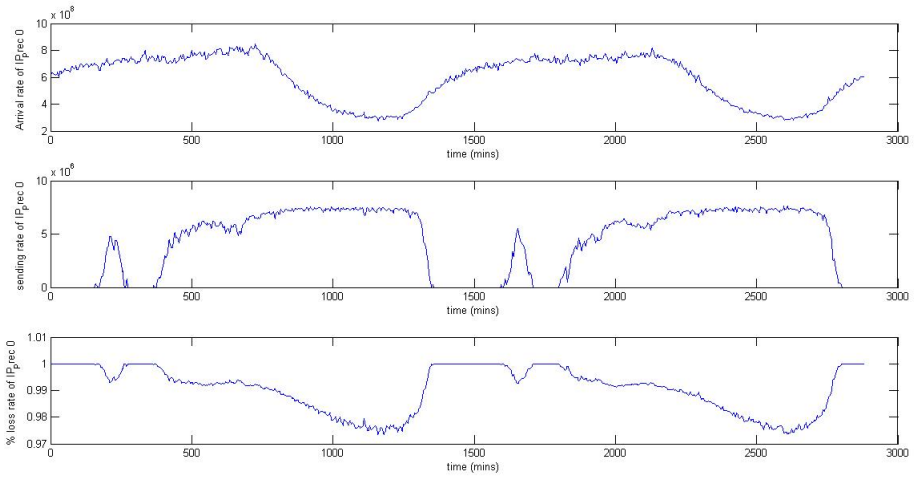


FIGURE 4.20: PQ:IP Precedence 1 - Flow behavior

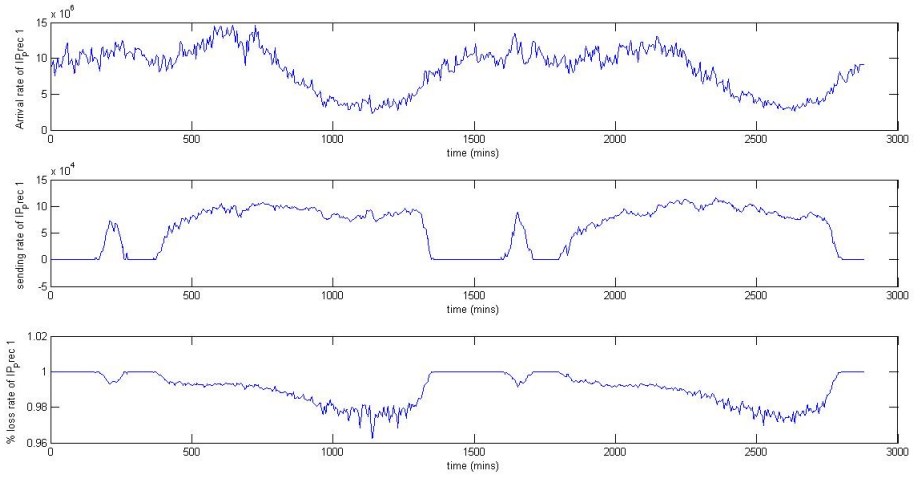


FIGURE 4.21: PQ:IP Precedence 3 - Flow behavior

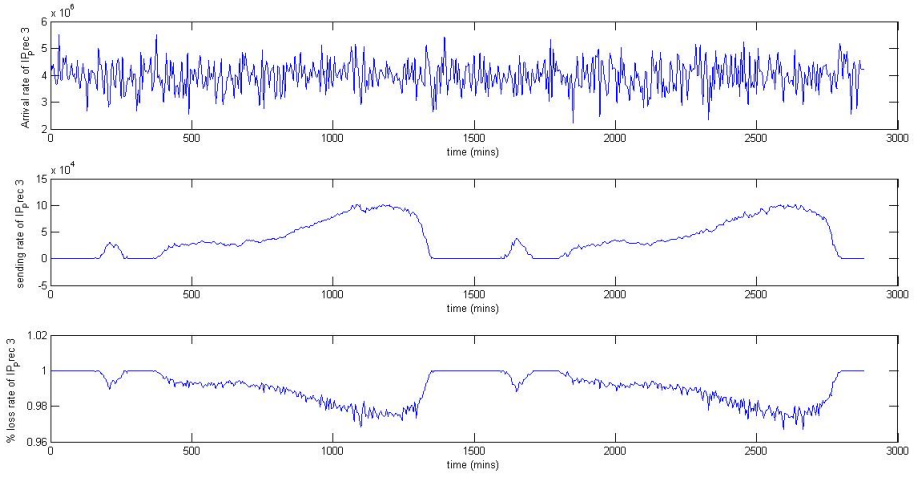


FIGURE 4.22: PQ:IP Precedence 2 - Flow behavior

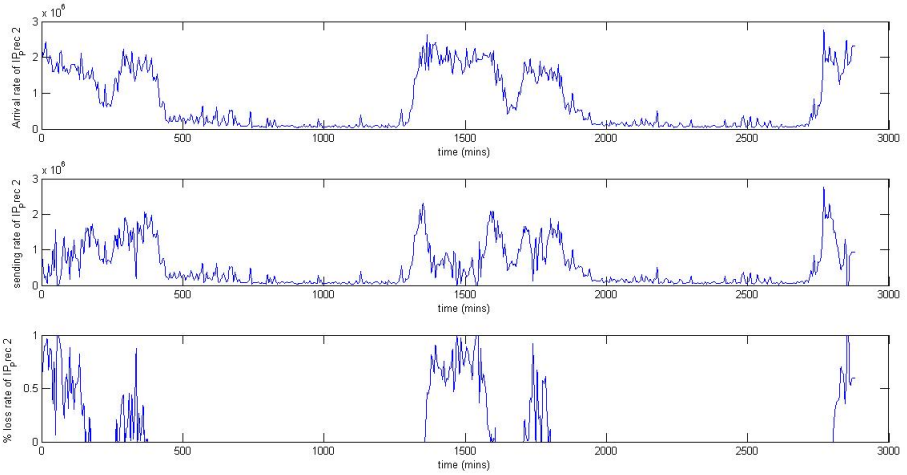


FIGURE 4.23: PQ:IP Precedence 4 - Flow behavior

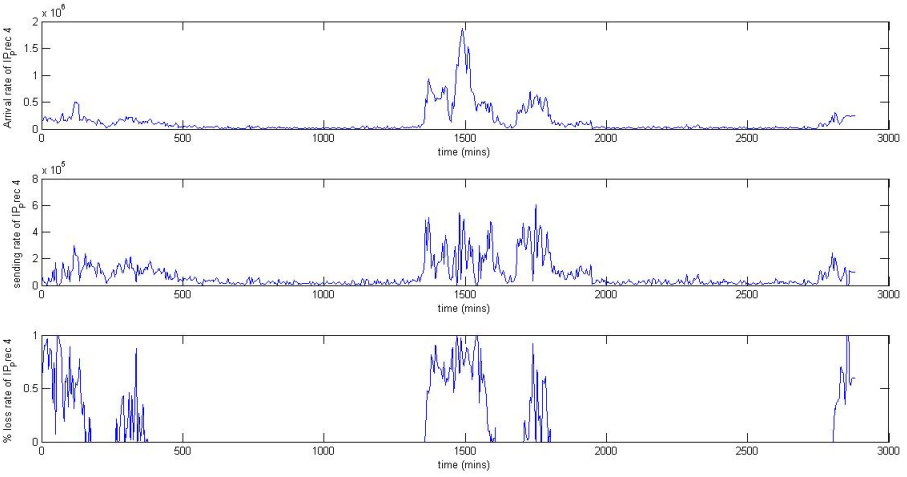


FIGURE 4.24: PQ:IP Precedence 6 - Flow behavior

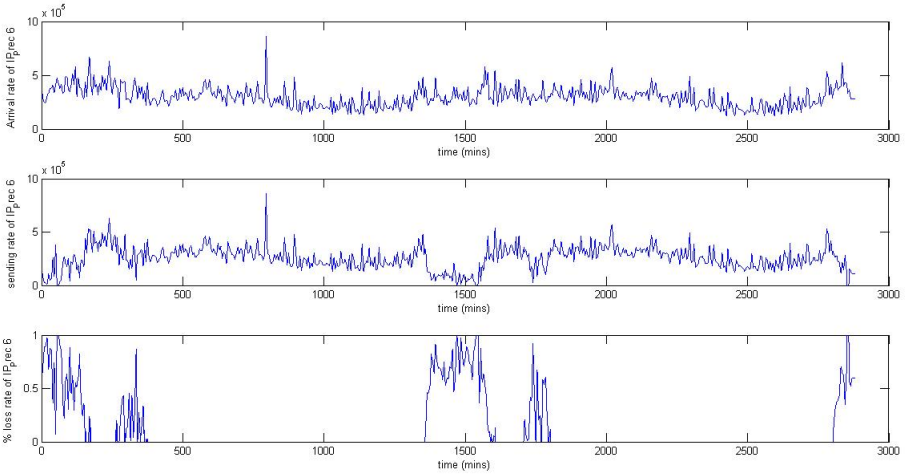


FIGURE 4.25: PQ:IP Precedence 7 - Flow behavior

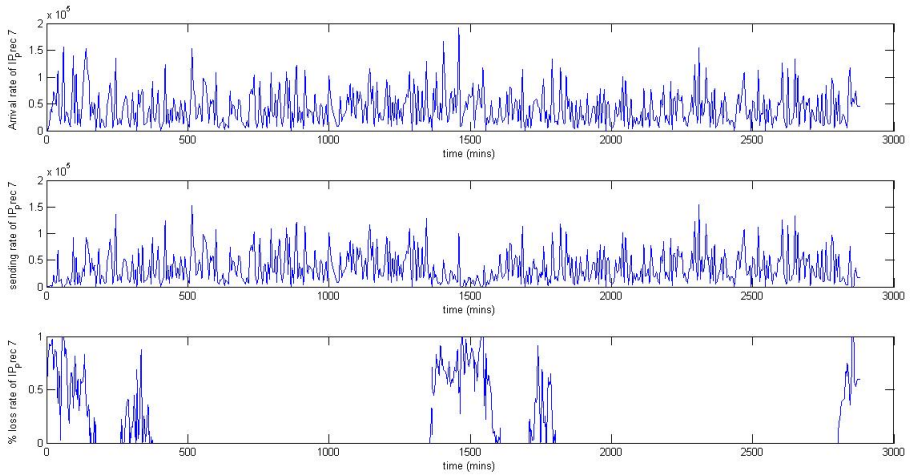


FIGURE 4.26: PQ:IP Precedence 5 - Flow behavior

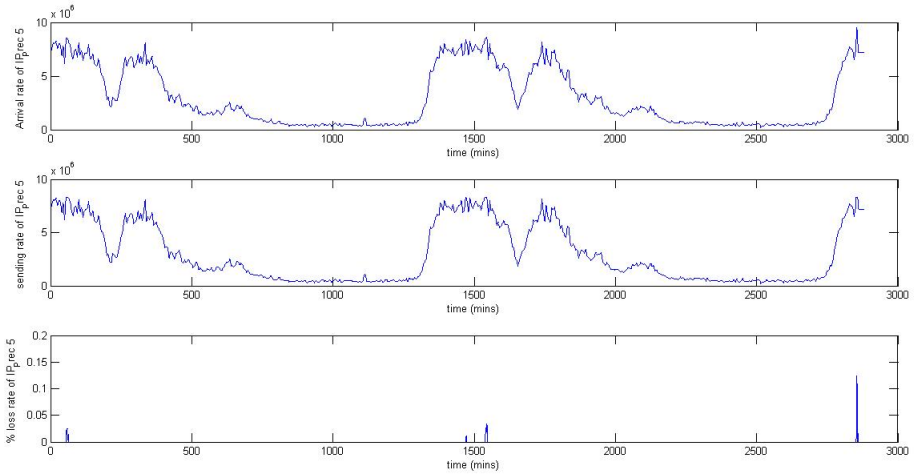


FIGURE 4.27: PQ: Queue Sizes

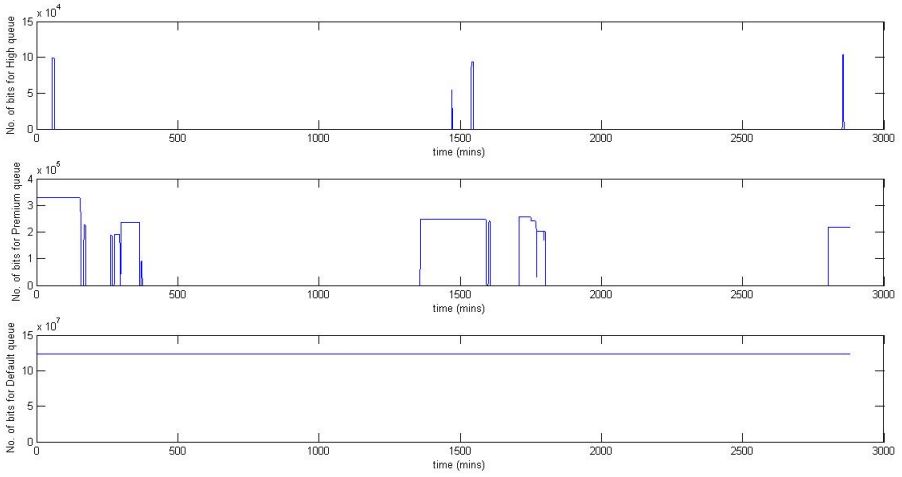


FIGURE 4.28: PQ:TCP Flow behavior for Video Streaming Service

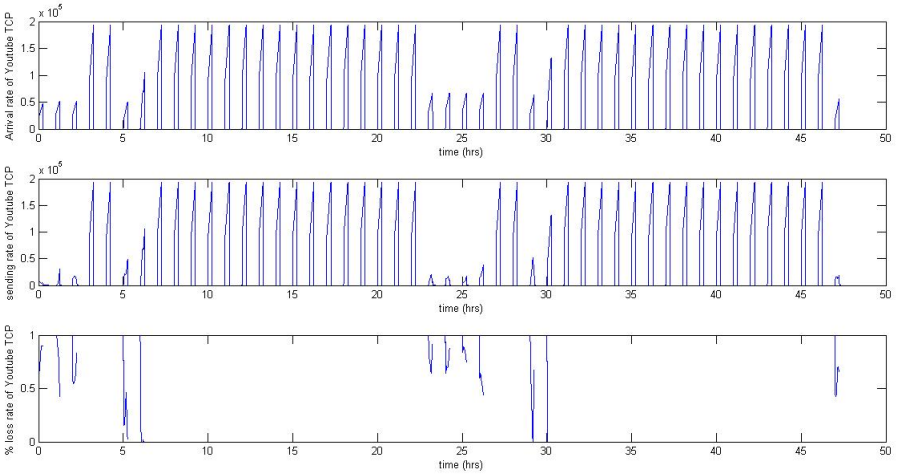


FIGURE 4.29: PQ:Queuing delay of TCP flow for Video Streaming Service

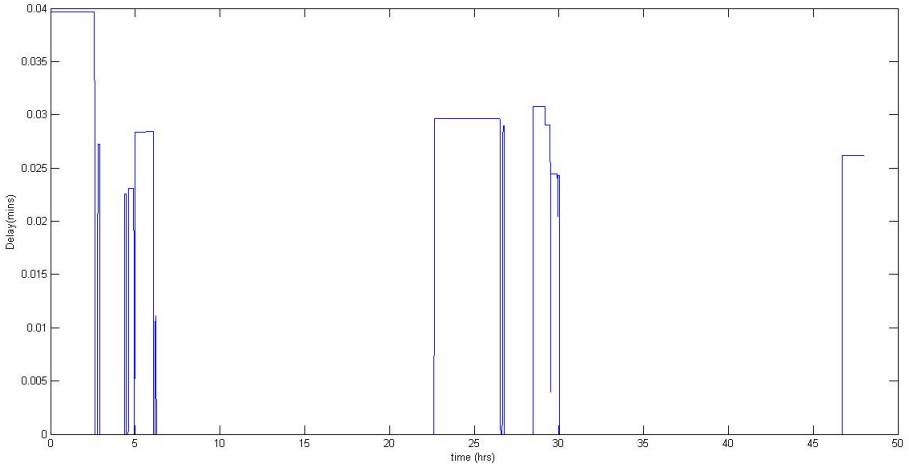


FIGURE 4.30: PQ:PVQ and Packet Loss rate %

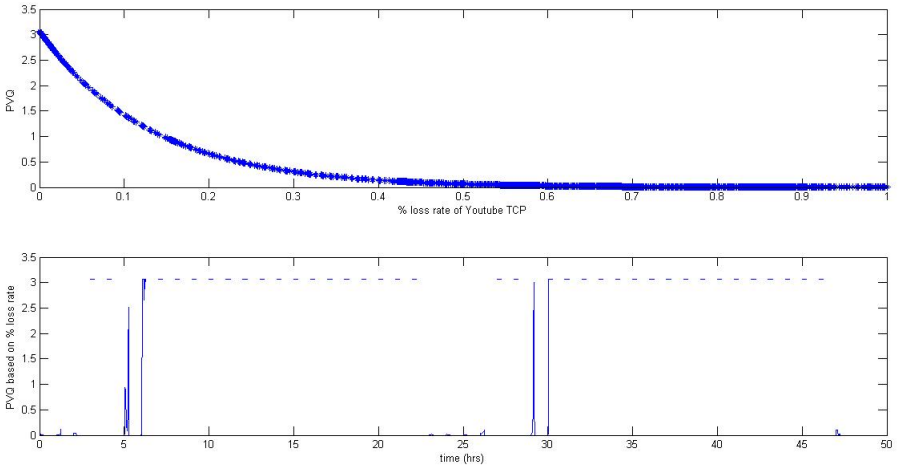


FIGURE 4.31: PQ:PSNR and Packet Loss rate %

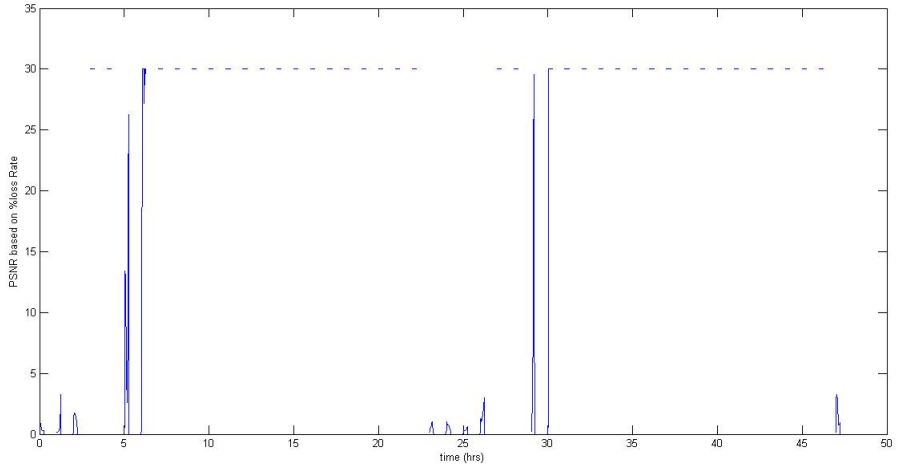


FIGURE 4.32: PQ:PSNR and Video Bit Rate

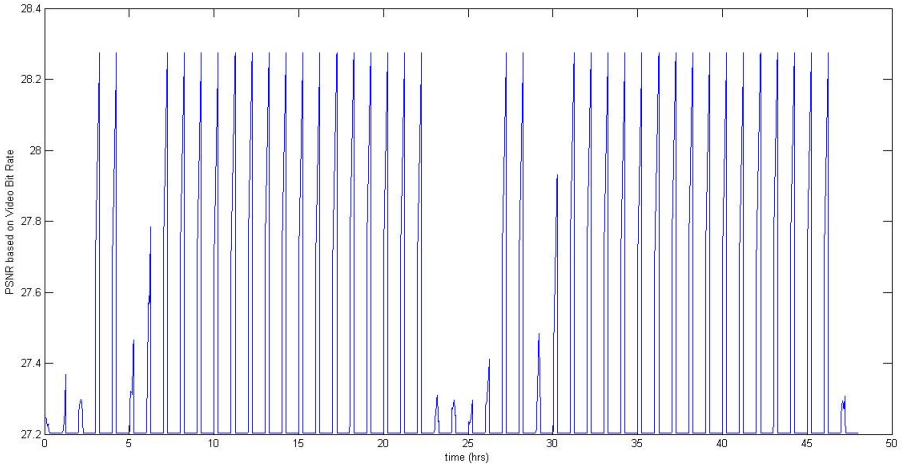


FIGURE 4.33: PQ:PVQ and PSNR(Packet Loss rate %)

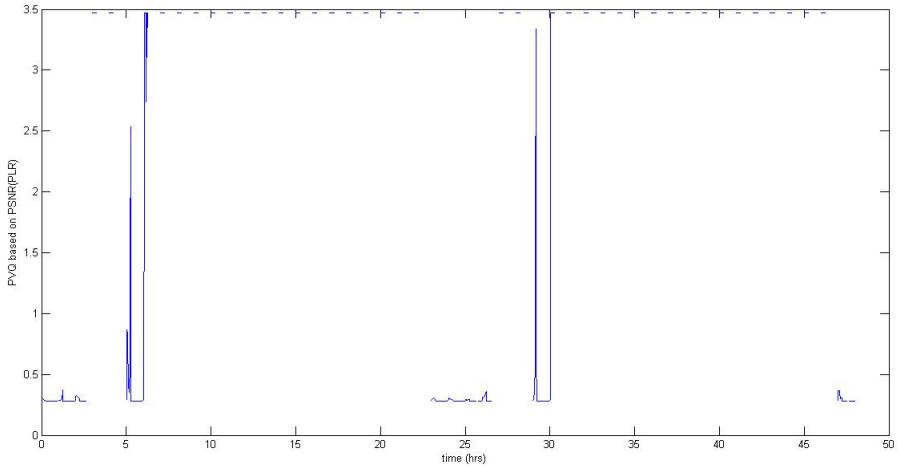
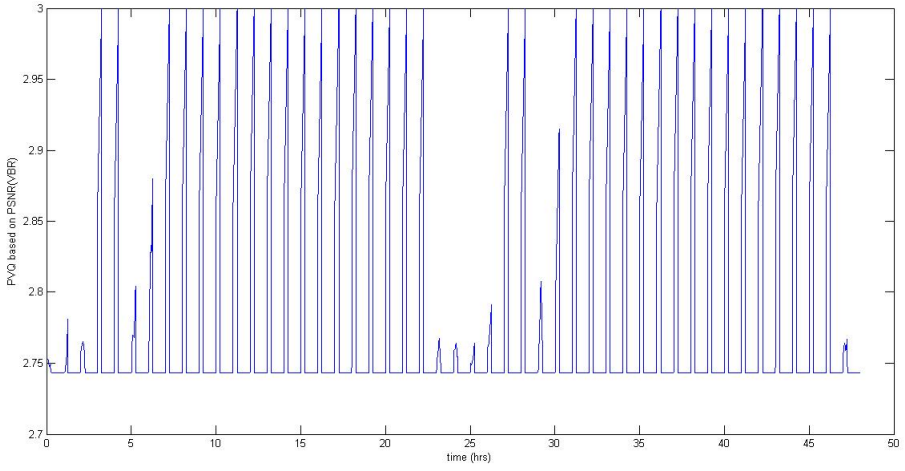


FIGURE 4.34: PQ:PVQ and PSNR(Video Bit Rate)



Chapter 5

Conclusion

5.1 Concluding Remarks and Future work

This is a first step towards setting up mathematical model for QoS. To summarise, the queuing delays are frequent(more number of drop events) in priority queuing compared to MDRR and therefore QoE is worse with priority queuing.But with respect to packet losses and video bit rate, Priority queuing has a better QoE.Packet losses for lower priority flows are more in priority queuing than in MDRR. This is quite useful for Telecom Italia when it has to add a new customer to this network so that it can estimate the QoE of the service before implementing it in real scenario.

By changing the weight (% of available bandwidth) for Premium and Default queues, packet losses for lower priority flows could be reduced. The future work aims to compare and analyse for different combinations of weights for Telecom Italia's MDRR.Also to optimise and dynamically change the weights for each of the queues by using feedback of QoE measure.

Bibliography

[1] *Packet level model*, http://www.omnetpp.org/.

[2] *Ip quality of service*, Srinivas Vegesna , Cisco Press, 2000.

[3] M D Di Benedetto, A Di Loreto, A D Innocenzo, and T Ionta, *Modeling of traffic congestion and re-routing in a service provider network*, (2013).

[4] Stephan Bohacek, P Hespanha, and Junsoo Lee, *A Hybrid Systems Modeling Framework for Fast and Accurate Simulation of Data Communication Networks [Extended Version]*.

[5] Cisco, *Ip precedence for video streaming*, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoS-SRND-Book/QoSIntro.html.

[6] ———, *Quality of service networking*, http://docwiki.cisco.com/wiki/QualityofServiceNetworking.

[7] Marcus Eckert, Thomas Martin Knoll, and Florian Schlegel, *Advanced MOS calculation for network based QoE Estimation of TCP streamed Video Services*, 2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS) (2013), 1–9.

[8] Khalil Rehman Laghari, Omneya Issa, Filippo Speranza, and Tiago H Falk, *Quality-of-Experience Perception for Video Streaming Services : Preliminary Subjective and Objective Results*.

[9] Vishal Misra, Wei-bo Gong, and Don Towsley, *Stochastic Differential Equation Modeling and Analysis of TCP-Windowsize Behavior*.

[10] Vishal Misra and Don Towsley, *Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED*, (2000).

[11] Ricky K. P. Mok, Edmond W. W. Chan, and Rocky K. C. Chang, *Measuring the quality of experience of HTTP video streaming*, 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops **1** (2011), 485–492.

[12] F Valentini, M Pratesi, and F Santucci, *Analisi dei Flussi dei Servizi di Video Streaming : il Caso di YouTube*, (1969).