ENHANCEMENT OF CONGESTION CONTROL PROTOCOL FOR TCP

A MAJOR PROJECT SUBMITTED TO
FACULTY OF TECHNOLOGY
UNIVERSITY OF DELHI

In the partial fulfillment of the requirements
for the award of the degree of

MASTER OF ENGINEERING
IN
COMPUTER TECHNOLOY & APPLICATONS

Submitted by:

RAVI GUPTA
Roll No. 8513

Under the Guidance of

Dr. S.K.Saxena
Department of Computer Engineering

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
UNIVERSITY OF DELHI

ABSTRACT

In heterogeneous networks such as today's Internet, the differentiated services architecture promises to provide congestion control through scalable service differentiation for TCP flows. This thesis proposes the use of TCP aware network based packet marking which in conjunction with differential packet dropping, is a powerful way to improve the performance of buffer management for congestion control in a TCP network. We extend the notion of TCP Friendly packet marking proposed recently and apply it to improve the performance of traditional best effort services. The TCP aware use of deterministic packet marking at the network edge allows us to protect selected TCP packets from suffering loss. In this thesis we list out the issues related to designing a TCP-friendly marker and propose an intelligent marker to address those issues. The major benefits of our markers are its simplicity, least sensitivity to parameters and transparency to the end hosts. We show baseline results which illustrate that the performance gains are considerable (orders of magnitude) when compared to stateful packet dropping algorithms like FRED [1] and even when TCP SACK [2].

ACKNOWLEDGEMENT

I take immense pleasure while presenting Enhancement of Congestion Control Protocol for TCP, as my final semester project.

This project report, as we see today in its present shape is an outcome of persistent effort and a great deal of dedication over a period of 6 months and has drawn intellectual and moral support from various people within the institution.

I am extremely indebted to my honorable guide Dr. S. K. Saxena for his invaluable support and guidance throughout the development of this project. It is a matter of great pride for me to have worked under him, which in itself was a source of inspiration for me to complete the project with great enthusiasm and determination.

I take this opportunity to thank all members of Computer Engineering Department for their valuable help in this project.

Finally, but immensely, I would like to thank to all my friends at DCE who helped me at various levels in this project.

RAVI GUPTA
08/CTA/04

CERTIFICATE

This is to certify that the project report titled**" Enhancement of Congestion Control Protocol for TCP"** has been submitted by **Ravi Gupta** student of final semester M.E. Computer technology & Applications of Delhi College Of Engineering as a part of his final semester project. The project was carried under the guidance of **Dr. S. K. Saxena** in the academic year 2006.

Dr. S. K. Saxena
Project Guide
Department of Computer Engineering
Delhi College of Engineering

Abstract……..……..2

Acknowledgement……..3

Certificate……4

1. INTRODUCTION 8

1.1 Solutions for Preventing Congestion Collapse in the Network 9

1.1.1 End System Based Approaches 10

1.1.2 Network Based Approaches 12

1.1.3 Summary 13

1.2 Fairness and Compliance in the Network 14

1.2.1 Fairness 14

1.2.2 Protocol Conformance 15

1.2.3 Summary 17

1.3 Contribution of this Thesis 18

1.3.1 Thesis Objective 18

1.3.2 Quality of Service 19

1.4 Organization of the Thesis 20

2. BACKGROUND 20

2.1 End System Based Mechanisms for Preventing Congestion Collapse 20

2.1.1 TCP and its Variants: Congestion Avoidance and Control 21

2.1.2 TCP and Drop Tail Queues 21

2.1.3 Rate Based Proposals for End-System Based Congestion Control 22

2.1.4 TCP Pacing: Solution for Reducing Burstiness of TCP 23

2.2 Network Based Mechanisms for Congestion Avoidance 23

2.3 Fairness 24

2.3.1 Oblivious Schemes for Providing Fairness in the Network 24

2.3.2 Non-Oblivious Schemes for Providing Fairness in the Network 24

2.4 Compliance in the Network 25

2.4.1 End-System Based Schemes for Compliance in the Network 25
2.4.2 Network Based Schemes for Compliance in the Network 26
2.5 TCP – Friendly Marker Based Approach 27
2.5.1 TCP-Friendly Best Effort Services 27
2.5.2 Performance Problems of TCP Over Assured Service 29
2.5.3 TCP-Friendly Packet Marker 30
2.6 The "Expected Capacity" Framework 32
2.6.1 Overview 32
2.6.2 Location of Profile Meters in the Network 32
2.6.3 A spectrum of services 34
2.6.4 Provisioning with Statistical Assurance 34
2.6.5 Receiver-Controlled Scheme 35
2.6.5.1 Difference Between Sender-Based Control and Receiver-Based Control 35
2.6.6. "Expected Capacity" for Bulk Data Transfers 36
2.6.7 TCP Rate Adjustment in the Current Internet 37
2.7 Differential Dropping in the Routers: RIO 38
2.7.1 RED Algorithm 38
2.7.2 Twin Algorithms in RIO 39
2.7.3 Profile Meters for Bulk Data Transfers: TSW Tagger 40
2.7.4 Difficulties in Designing RIO-TSW 41
3. QUALITY OF SERVICE (QOS) MODELS 42
3.1 Absolute Differentiation vs. Relative Differentiation 43
3.2 Service Models 44
3.2.1 The Integrated Services (IntServ) Model 44
3.2.1.1 Guaranteed Services 45
3.2.1.2 Controlled Services 45
3.2.2 The Differentiated Services (DiffServ) Model 46
3.2.2.1 Expedited Forwarding PHB 50
3.2.2.1 Assured Forwarding (AF) PHB 51
3.2.3 The Rate Proportional Differentiation (RPD) Model 52
3.3 Taxonomy on Service Models 53
3.4 Other QoS related issues 54
3.4.1 Traffic Engineering and Constraint Based Routing 54

3.4.2 Traffic Modeling 55
3.5 Summary 55

4. INTELLIGENT MARKER 56

4.1 Design Issues 56
4.2 The Marking Algorithm 57

5. IMPLEMENTATION DETAILS 59

5.1 The Scenario 60
5.2 Simulation Parameters 61
5.3 Results and Analysis 62
5.3.1 Assured Service for Aggregates with Different Target Rates 62
5.3.1.1 Analysis 63
5.3.2 Effect of Different RTTs 65
5.3.2.1 Analysis 65
5.3.3 Effect of different window sizes 66
5.3.3.1 Analysis 67
6. INFERENCE AND FUTURE WORK 68

References 69

1. INTRODUCTION

Over the years as the Internet has evolved TCP has formed the backbone of its stability. However, a decade ago the present Internet suffered from a severe congestion control problem, which was called the “Internet meltdown". To prevent such a situation Jacobson [3] proposed the congestion avoidance and control mechanisms for TCP which has subsequently become the de-facto transport protocol for the Internet.

However as the application needs changed newer rate control schemes were proposed. As such we now have an Internet which operates with a spectrum of congestion control schemes, even though TCP remains the most widely used transport protocol. In [3] the authors have argued that these new congestion control schemes can lead to a new congestion collapse and pose the problem of protocol conformance (wherein selfish schemes get an unfavorable share of bandwidth in comparison to TCP).

Though end-system based congestion control mechanisms have helped prevent Internet meltdown they are not sufficient to provide good service under all circumstances. Specifically network and end-user performance may degrade in presence of Drop Tail queues and different rate control schemes. Also end system based solutions constrain the choices of flow control protocols which might be available to any application. Towards addressing these issues, router (or network) based schemes like Active Queue Management (or AQM) has been proposed as a complement to end-system based congestion control schemes.

However these AQM proposals are beset with configuration problems and also require up gradation of the network (i.e. each bottleneck must have the AQM enabled). As a result of these implementation drawbacks, the Internet still operates with Drop Tail queues. Considering that AQM schemes are still to be widely deployed on the Internet and presence of different congestion control schemes, in this thesis we look at deployable end system and network based solutions to improve fairness and protocol conformance in the network. In the following sections we first present the various solutions for preventing congestion collapse, providing fairness and protocol conformance in the network. Specifically we look at the network and end point solutions and discuss their current deployment in the network. Thereafter we present our proposals for protocol conformance and improving fairness through the introduction of an intelligent marking algorithm.

1.1 Solutions for Preventing Congestion Collapse in the Network

The Internet protocol architecture is based on a connectionless end-to-end packet service using the IP protocol. The advantages of its connectionless design, flexibility and robustness have been amply demonstrated. However, these advantages are not without cost: careful design is required to provide good service under heavy load. In fact, lack of attention to the dynamics of packet forwarding can result in severe service degradation or "Internet meltdown".

As a result of this meltdown considerable research has been done on Internet dynamics and many solutions have been suggested to avoid it. These proposals can be broadly classified into two categories a) end-system based solutions, e.g. TCP and other congestion control schemes and b) network based solution. In this section we briefly discuss these proposals, their advantages and disadvantages and their current deployment in the Internet.

1.1.1 End System Based Approaches

The end-system based solutions consist of source or receiver based congestion control schemes. These schemes try to avoid congestion in the network by cutting down their transmission rate, whenever congestion is detected. The original fix for the congestion collapse (or Internet meltdown) proposed by Jacobson in 1988 is one such scheme. In particular, Jacobson proposed the congestion avoidance and control features in TCP and since then TCP has been the mainstay of the Internet.

These end-system based solutions can operate with and without network support. In absence of network support the network employs simple queuing at the routers in which the packets are admitted till the queue has space. This queuing policy is called Drop Tail. Though simple to implement, Drop Tail queue do not try to manage congestion in the network, in fact it is left to the end-system based application.

Though TCP has served the Internet community well it is known to suffer from a number of phenomena which limits its effectiveness when operated over a network of Drop Tail queues. The main problem which degrades TCP and network performance are: synchronization of congestion windows (or correspondingly the loss instances) causing alternate overloading and under-loading of the bottleneck [3]; phase effects wherein a certain section of flows face recurrent losses [3]; unfairness to flows with higher RTTs [7]; bias against bursty traffic [4] ; delays and losses due to the bursty nature of TCP traffic [3, 4].

The tail-drop discipline allows queue to maintain a full status for long periods of time. This is because Drop Tail signals congestion only when the queue has become full. If the queue is full, an arriving burst will cause multiple packets (from same or different flows) to be dropped causing global synchronization . This synchronization can be attributed to two reasons: (1) the sliding window flow control of the TCP, which produces bursts of packets and (2) the Drop Tail queue at the bottleneck, which drops all packets when the

buffer is full [3]. Synchronization of windows and loss events for flows sharing common links causes alternating periods of overload and under-load thereby leading to inefficient resource utilization. In some situations Drop Tail queuing allows a single connection or a few flows to monopolize queue space, preventing other connections from getting room in the queue. This "lock-out" phenomenon is often the result of synchronization .

Phase effects refer to conditions where in the bandwidth-delay product of the path of a flow is not an integral multiple of the packet size [2]. Phase effects cause a specific section of competing flows to experience recurrent drops causing unfair distribution of bandwidth and increased latency. Phase effects are manifested in the network preferentially dropping packets from a specific subset of flows thereby reducing their throughput. Drop Tail queues suffer from a problem called, "full queues", which implies that Drop Tail queuing maintains sustained long or full queues. The sliding window protocol of TCP and the persistent full queues often results in burst losses. These burst losses causes Drop Tail queues to differentiate against TCP like schemes [3]. It has been widely shown that TCP can recover well from a single packet loss but with burst losses it often times out [5]. Consequently, these burst losses also increase the delays. It has also been reported that these burst losses are the primary reason for the bias against flows with longer RTT [3].

Drop Tail queues also do not protect flows. As noted earlie because of synchronization Drop Tail queues can let some flows monopolize the buffer space. Also, given that there are various congestion control schemes in the network, by not differentiating amongst flows, Drop Tail queues allow aggressive sources to get more bandwidth. Flows which do not react to congestion indications will push the responsive flows out of the queue and will always take up bandwidth worth their transmission rate [8]. Thus by introducing burst losses and by not protecting flows, Drop Tail queues aggravate the problem of unfair equilibrium rate allocations in the network.

1.1.2 Network Based Approaches

Though the end-system based congestion avoidance and control mechanisms are necessary and powerful, they are not sufficient to provide good service under all circumstances. Primarily there is a limit to how much control that can be accomplished from the end of the network. Specifically these problems were highlighted in the previous section and can be chiefly attributed to the full queues and lock-out behavior of the Drop Tail queues. Thus some mechanisms are needed in the routers to complement the endpoint congestion avoidance mechanisms.

Active Queue Management (AQM) was suggested as a pro-active way of managing queue at the bottleneck router. The pro-activeness was defined to be able to drop few packets before the queue gets full thereby signaling sources to cut their rates on account of impending congestion. This in turn help solved the problem of full queues. The solution to the full queues problem implied that there would be space in the queue to enque packets which consequently solved the lock-out problem of Drop Tail queues.

Random Early Drop (or RED) [3] was one such AQM proposal wherein the authors suggested to probabilistically drop packet when the queue size gets above a certain threshold. This probabilistic dropping distributed losses over time thus making them appear independent. It also introduced randomization at the bottleneck which in turn broke synchronization amongst flows and improved network performance. Also, by sending early congestion signal (by dropping a packet before the queue actually gets full) helped manage queues efficiently and also provided space to accommodate bursts. Thus RED avoids burst losses, synchronization, reduces the bias against long RTT flows and prevents timeouts.

However it's been almost a decade since the RED proposal but the Internet still operates with Drop Tail queues. This can be explained by lack of guidelines to set RED. Studies have shown that if not properly configured the performance of TCP with RED queues

may even be worse than those with Drop Tail queues. Specifically, in [5, 9] the authors show that the probability of consecutive drops increases with RED queues. Though there have been some studies on how to configure RED these works attempt to configure only one or a set of parameters and as such have not found much favor with the network operators [6, 8]. Besides RED there have been other AQM proposals which have fewer parameters to configure and crisper guidelines for setting them [7]. But in-spite of numerous AQM propositions the network still operates with Drop Tail queues and consequently the problems of TCP and Drop Tail queues exist to this day.

1.1.3 Summary

The policies outlined for preventing congestion collapse require either end system support in form of congestion control scheme or router based schemes like AQMs. However there is a limit to the control which can be achieved by using the end-point congestion schemes. In absence of network control we have seen that the flows are subjected to burst losses and can get synchronized which in turn limits the effectiveness of TCP. Further, end-point congestion control scheme do not protect flows, on the contrary they allow some flows to monopolize the buffer space.

Network control for preventing congestion collapse was envisioned in form of Active Queue Management. RED was one such proposal which absorbed bursts by probabilistically enqueing packets. This introduced randomness at the bottleneck and helped avoid synchronization of flows. However RED's performance is highly sensitive to its parameter configuration, so much so that at times the performance of Drop Tail queues might be better than RED queues. This problem is further compounded by the lack of guidelines for setting these parameters. As such the network still operates with Drop Tail queues.

To summarize, due to configuration and implementation problems with AQM the Internet still operates with Drop Tail queues. As a result, the problems of bursts losses, flow synchronization, bias against flows with longer RTT and manipulation of buffer space by

selfish and unresponsive flows persist. These problems in turn limit the effectiveness of TCP and degrade the performance of the network.

1.2 Fairness and Compliance in the Network

1.2.1 Fairness

Fairness can be defined in a number of ways but its essence in each of these definitions is that it is some measure of the distribution of the allocated rate amongst users. Fairness is related not only to the network but also to the end-system's congestion control scheme. Often the end-system's objective is to be fair to the other competing user's while the network's objective is that it does not arbitrarily penalize or differentiate amongst various competing user.

Traditionally, the Internet has relied on the "end-to-end" congestion control model like TCP (or alternate transport protocols) where end users choose a rate control scheme, and the network merely drops or marks packets during congestion as a method to convey the penalty or price. One implication of this model is that end-systems are free to choose any rate control scheme.

The network on the other hand seldom differentiates between flows. It drops (or marks) packets obliviously, i.e. drop packets whenever there is no space in the router queue or if the router queue length crosses a certain threshold. Drop Tail queuing, RED and many of RED's variants can be classified as oblivious queuing disciplines. This oblivious dropping coupled with the flexibility to end system to chose a rate control scheme makes the problem of providing fair rate allocations to all users hard.

The “fair" equilibrium allocations in an oblivious network therefore depend upon the utility functions chosen freely by users. These equilibrium allocations, though fair under Kelly's framework, might be unfair from network perspective.

Moreover the fair rate allocation problem is further compounded by the fact that there is no single definition of fairness. The two most common definitions of fairness are *max-min* and *proportional* fairness [4]. In *max-min* fairness criteria the objective is to maximize the minimum unsatisfied rate allocations. Thus given the same network conditions, two competing flows should get equal share of the bottleneck. On the other hand, in *proportional* fairness the rate allocations are in proportion to the network resources being used. But all the same, a more general definition of fairness, (p,a) fairness is defined in [5]. Thus irrespective of user's rate control schemes it is for the network provider to decide the criteria for allocating resources amongst users.

Over the years equal allocations and Max-Min fairness [5] have formed the network's view of fair allocations. As such, AQM schemes and schedulers deployed at every bottleneck have been used to enforce conformance with these definitions, by penalizing misbehaving users. For example, CHOKe tries to enforce Max-Min fairness [5] across the network. Similarly fair queuing and it's variant have also been used to provide Max-Min fairness. Finally to summarize, any arbitrary fairness objective cannot be achieved by AQM schemes though they could be arrived at by use of schedulers throughout the network.

1.2.2 Protocol Conformance

The congestion control scheme in TCP has been the focus of numerous studies and consequently gone through lots of changes. These changes were also motivated by varying needs of the applications using the Internet. As such, even though TCP remains the most widely used protocol; we have now a spectrum of congestion control schemes. In [3] the authors show that absence of end-to-end congestion control schemes or presence of selfish users could not only lead to TCP being beaten down but also may even result in congestion collapse.

This thus represents the problem of protocol conformance. In absence of compliance to a set of protocols, for example TCP, we might be faced with the problem of TCP flows

being singled out and gets rates which are (significantly) less than their fair share. This problem is further highlighted in presence of unresponsive flows. (Flows which do not cut down their rates upon receipt of congestion indication are called unresponsive flows.) These unresponsive flows can shut-out TCP because on occurrence of congestion, TCP will cut its rate and the unresponsive flows will step in to take the available bandwidth. A similar problem is posed by responsive selfish flows (i.e. flows who react to congestion indication but are selfish as compared to TCP) in the network. Specifically, these flows could have a rate increase policy which is faster than TCP and some flows could have a rate decrease policy slower than that of TCP.

Given that TCP is the most widely used transport protocol, Floyd et al proposed the guidelines for managing and designing new congestion control schemes such that they were friendly to TCP. A flow is deemed TCP-Friendly if its sending rate does not exceed that of a conformant TCP flow in same circumstances. This TCP-Friendly definition can further be loosened to the following relationship between the sending rate, x, and loss rate, p: This TCP Friendliness can also be understood as protocol compliance, as all flows try to be conformant to TCP.

TCP-Friendliness is the criteria not only for safeguarding TCP flows but also for enforcing some kind of fairness in the network. (TCP-Friendliness ensures Minimum Potential Delay Fairness across the network [4]) Further, we could easily expand TCP Friendliness definition to encompass a larger range of rate control scheme which can be done by relaxing the relationship between the sending and loss rates. However, enforcing TCP Friendliness on the network remains a challenging question. In [3] the authors argue that router-based mechanisms are needed to administer TCP-Friendliness. Other than these router based schemes, another way of enforcing TCP Friendliness could be design of end-point congestion control algorithms which are TCP-Friendly.

In [9] authors have proposed a general class of TCP-Friendly congestion control schemes. Though these proposals are encouraging they solve only a part of the TCP-Friendliness problem because the end-users may willfully choose to ignore these TCP-

Friendly guidelines. As such it becomes imperative to have router-based mechanisms for enforcing TCP-Friendliness. In [3] the authors point out using per-flow scheduling or pricing mechanisms for enforcing TCP-Friendliness. However, till date to the best of our knowledge, no such per-flow scheduling or pricing mechanisms have been proposed or deployed to achieve TCP-friendliness on the network.

1.2.3 Summary

From the above discussion it follows that protocol conformance and fairness are very closely related. Protocol conformance guarantees a certain kind of fairness, for example TCP-Friendliness will result in minimum potential delay fairness across the network [4]. Similarly any fairness definition can always be translated to another protocol conformance.

Traditionally Max-Min fairness has formed the network's definition of fairness. This definition aims to provide equal allocations to different flows. However TCP allocates rate in proportion to the round-trip times (RTT) of the flows and loss rate. This then stands in contradiction to the network's traditional fairness goals. Thus AQM and scheduling disciplines which enforce Max-Min fairness do nothing to enforce TCP-Friendliness.

Also these different schemes for providing network-wide fairness have their own drawbacks:

- We would need AQM or scheduler support throughout the network. This implies that we will have to make changes in the core.
- TCP-Friendly criteria constrain the choice of end-point congestion control schemes for users. This might also infringe with the requirements of different protocols as these needs might not always be satisfied with TCP.

- Both AQM/Schedulers and TCP-Friendliness cannot provide a broad range of fairness criteria in the network.

1.3 Contribution of this Thesis

There is a distinct need to move towards providing service differentiation (diff-serv) in the Internet. The current best-effort model does not provide any means of preferentially treating traffic from customers who are willing to pay more. The differentiated service model uses a combination of network edge elements (traffic conditioners) and network core elements (per-hop behaviors or PHBs) to achieve service differentiation [1, 2].

In this thesis, we consider a simplified form of a better than-best-effort service called the "assured service"[6]. The building blocks of this service include a traffic marker at the edge of the domain, and a differentiated dropping algorithm in the network interior. The traffic marker marks packets as "IN" or "OUT" (corresponding to the two "colors") depending upon the service level agreement (SLA). An example of a differentiated dropping algorithm is "RIO" - a variant of the RED (Random Early Gateway) algorithm . The RIO algorithm ("RED with In and Out") uses the same RED algorithm for "IN" packets and "OUT" packets, albeit with a different set of parameters for each. In particular, the OUT packets are preferentially dropped upon evidence of congestion at the bottleneck before the IN packets. For example, a minimum rate of packets could be marked as "IN" in expectation of a minimum "assured rate" because the "IN" packets would have high probability of delivery.

1.3.1 Thesis Objective

TSW profile meters (TSW–TC) [7] have two components: a rate estimator that estimates average sending rate over a time window (Tw), and a marker that tags packets as in-profile or out of profile .There are two approaches to use TSW profile meter: in the first approach, it remembers a relatively long past history (Tw is large); in the second approach, it remembers a relatively short past history (Tw = RTT).The problem

associated with the first approach is that it cannot reflect well the traffic dynamics of TCP. The drawback of second approach is that the average rate of packets that are marked as in-profile will be much more than the target rate in the under-subscribed scenario (i.e., when the actual throughput attainable is significantly higher than the target rate).

Recent measurements across the transatlantic links have shown TCP flows being in majority with almost 95% of the byte share [8].TCP flows due to its congestion avoidance and slow start mechanisms [10] are much more sensitive to congestion, especially to multiple drops. Also, the TCP parameters- like send and receive window sizes if not tuned appropriately might affect the flow throughputs. Hence, providing AS to TCP flows has been an active research issue .It assumes more significance in the present day world, with more and more non-TCP flows flooding the networks, which make the TCP flows vulnerable .Thus, there is a need for designing intelligent TCP-friendly marking algorithms, which take care of the TCP dynamics as well.

1.3.2 Quality of Service

Quality of Service is the classification of packets for the purpose of treating certain classes or flows of packets in a particular way compared to other packets. It means making unpredictable data delivery service of the currently best effort Internet predictable in terms of bandwidth, jitter and propagation delay. There is a need for some service differentiation in the present Internet architecture to provide QoS.

The class of assured services (AS) [7] is intended to give the customer the assurance of a minimum throughput, called the target rate, even during periods of congestion, while allowing it to consume, in some fair manner, the remaining bandwidth when the network load is low. The AS architecture relies on packet marking mechanism, performed by the traffic conditioner (TC), at the edge routers, and queue management mechanism at the core routers, to realize the above objectives. The TC that is used at the edge router for marking the packets as in-profile and out-of-profile can be classified into two broad

categories: token bucket (TB) based and average rate estimator based, also called time sliding window (TSW) profile meter.In our work, we use the terms profile meter and TC interchangeably.

1.4 Organization of the Thesis

In this thesis we propose an intelligent TCP friendly marking algorithm for the TSW–TC. The rest of the thesis is organized as follows: Section 2 gives an overview of the related work on TCP friendly markers done so far. Section 3 explains the design issues and algorithm for intelligent marker in detail. Section 4 presents the simulation environment. Section 5 presents the results and their analysis for different cases. We conclude with our inferences and suggestions for future work in this area, in Section 6.

2. BACKGROUND

In this chapter we review the need of congestion control in the Internet and its objectives. Thereupon we will discuss the end-system based proposals for congestion control (e.g. TCP and its variants) as well as network based proposals i.e. AQM. This discussion on congestion control brings us to the question of how resources are shared between users, i.e. the problem of fairness. For these purposes we first review the definitions of fairness and then the various schemes for achieving fairness in the network. Finally we discuss the question of protocol compliance and flow control optimization framework.

2.1 End System Based Mechanisms for Preventing Congestion Collapse

In 1980s lack of attention to the dynamics of packet forwarding on the Internet resulted in severe service degradation or congestion collapse. Since this congestion collapse considerable research has been done on Internet dynamics and many solutions have been suggested to avoid it. The original fix for congestion collapse was provided by Van

Jacobson in 1988 as some modifications to TCP [4]. Ever since, TCP has been the backbone of the modern Internet.

2.1.1 TCP and its Variants: Congestion Avoidance and Control

TCP is a sliding window based transport protocol where the window is increased upon successful reception of acknowledgments (Ack). This ensures that TCP gradually probes and takes all the available bandwidth. However, this probing will result in a situation where the sender's sending rate exceeds the network capacity and at that point the network will drop the excess packets. These packet losses are construed as sign of congestion by the TCP and it reacts to it by cutting its rate (or decreasing window).

There have some other proposals to optimize the TCP and the most notable amongst those has been TCP SACK. Selective Acknowledgment (SACK) is a strategy wherein the receiver can inform the sender about all segments that have arrived successfully, so the sender needs to retransmit only the segments that have actually been lost.

Another proposal which merits mention is TCP Vegas . Unlike other TCP proposals which use packet loss or marking as a congestion notification TCP Vegas uses queueing delays to decipher congestion. TCP Vegas relies on the fact that during congestion the queues will build up at the bottlenecks and as such the queueing delay will increase. This increase in queueing delay is construed as a sign of congestion and TCP Vegas decreases its window by one packet, otherwise it increases its window linearly. Since the window increase and decrease in TCP Vegas is small it is most likely to converge to the optimal bandwidth.

2.1.2 TCP and Drop Tail Queues

TCP and other similar congestion control schemes result in bursty traffic. This burstiness can be attributed to three main reasons. One, when the window is increased the last two packets are sent back to back. Two, in presence of congestion on the reverse path the

Acks arrive back to back and as such the packets sent are back to back. And finally, when an Ack for a retransmitted packet arrives it might result in release of a previously stalled window which might in turn lead to back to back transmissions. In order to improve the performance of the network buffers are provided at the links to absorb these bursts.

A drawback of Drop Tail queues is that they don't protect flows, i.e. it allows for a section of flows to monopolize the entire bandwidth . This is especially important in the current Internet where the end system has a flexibility of choosing its congestion control scheme. This then raises the question of unfair sharing of the bottlenecks as the aggressive flows might corner a larger share of the bandwidth.

2.1.3 Rate Based Proposals for End-System Based Congestion Control

Jacobs [3] presents a scheme that uses the congestion control mechanisms of TCP, however, without retransmitting lost packets. In his scheme, the sender maintains a transmission window that is advanced based on the acknowledgments of the receiver, which are sent for each received packet. The sender then uses the window to calculate the appropriate transmission rate. Rejaie et al. presented an adaptation scheme called Rate Adaptation Protocol (RAP). Just as with TCP, every packet sent is acknowledged by the receivers and these acknowledgments the sender estimates the round trip delay. If no losses are detected, the sender periodically increases its transmission rate additively as a function of the estimated round trip delay. Upon detection of a loss the rate is reduced by half in a similar manner to TCP. However, this approach as well as the one presented in [4] does not consider the cases of severe losses that might lead to long recovery periods for TCP connections. Hence, the fairness of such an approach is not always guaranteed.

2.1.4 TCP Pacing: Solution for Reducing Burstiness of TCP

Sliding window based protocols like TCP often send packets in burst. As such the performance of sliding window protocols suffers on a network of Drop Tail queues. On the contrary, rate based schemes send out packets at regular intervals thus avoiding burst transmissions. However, since rate based schemes loosely observe the packet conservation principle they at times can be less responsive to network congestion. TCP Pacing [3] is a hybrid approach between window based schemes and rate based schemes. In pacing, packets to be sent in a window are spaced by RTT / cwnd. This spacing of packets avoids back to back transmissions and hence removes the burstiness of TCP.

Pacing was first suggested in [3] as a correction for the compression of acks due to cross traffic. Since then the concept of pacing has been applied to slow-start, after a packet loss and after an idling time in case of web traffic . In order to speed up web connections the authors in [6] suggest using pacing during the slow start as means for Ack clocking. Similar results have been reported in [6] where the authors show that performance of slow start can be improved by use of pacing. Pacing has also been suggested for improving TCP performance with asymmetry [8] and on high bandwidth delay product links . In [6] the authors evaluate pacing over the entire lifetime of TCP as a means for reducing queueing bottlenecks in wireless, high bandwidth delay networks. In [5] the authors have proposed a fast web protocol, WebTP which uses pacing during congestion avoidance phase.

2.2 Network Based Mechanisms for Congestion Avoidance

From the discussion in previous section it is clear that the TCP congestion avoidance mechanisms, while necessary and powerful, are not sufficient to provide good service in all circumstances. However, there is a limit to how much control can be accomplished from the edges of the network. Some mechanisms are needed in the routers to complement the endpoint congestion avoidance mechanisms.

2.3 Fairness

Though there are many definitions of fairness but its meaning in all definitions is that it represents the distribution of rates between users. Fairness is one of the most important considerations before network providers. This is because it represents how a network distributes rates between users such that the network does not penalize any user and more importantly can also use it to provide service differentiation.

2.3.1 Oblivious Schemes for Providing Fairness in the Network

End based techniques are insufficient to protect flows in the network and thereby provide fairness. Towards achieving this objective use of AQM at routers was proposed. RED [3] was the first significant AQM proposal. However as discussed in Section 2.2 RED cannot protect flows, especially when TCP flows in the cases where they compete with unresponsive flows [8]. Moreover since RED's control parameter are statically configured, i.e. the configuration does not change with time, RED's penalty function can be severe under low loads and insufficient with large multiplexing of flows [2,4]. This further constrains the fairness objectives which RED can achieve.

In summary barring CHOKe all oblivious schemes cannot protect TCP from misbehaving flows. However, all oblivious schemes are limited by the range of fairness criteria they can provide.

2.3.2 Non-Oblivious Schemes for Providing Fairness in the Network

From discussion in the previous section it is clear that in order to protect flows from misbehaving users we will need to assign marks not only on the basis of aggregate arrival rate to the bottleneck queue but also on individual flow arrival rates. Thus if we are monitoring individual flow rates to assign marks (or drops), the subsequent schemes are called non-oblivious schemes. Different ways have been suggested for monitoring individual flow's share in the bottleneck. One of these is explicit rate monitoring at every bottleneck, another method involves monitoring at one bottleneck and then sending the

rate information through some means (either in packet header or through specific control packets) or deciphering the rate through number of packets enqueued in the bottleneck queue. In this section we will discuss some non-oblivious network based schemes which use one of these methods for providing fairness in the network.

2.4 Compliance in the Network

Over the years the Internet growth has been well supplemented by various applications who have varying needs, especially for transport protocols. Initially all these applications relied on TCP but as the requirements of the applications changed TCP were no longer the only favored transport protocols and therefore a variety of flow control algorithms were proposed. These flow control algorithms can be mainly divided into three main classes a) TCP-Compatible flows b) unresponsive flows, i.e., flows that do not slow down when congestion occurs, and (c) flows that are responsive but are not TCP-compatible [2]. Yet another class of flow control algorithm use a mix of responsiveness and unresponsiveness. Specifically these algorithms decrease their rate on receiving a congestion indication but they also have a lower limit on transmission rate, i.e. they do not react to congestion indications when the sending rate is below this limit.

2.4.1 End-System Based Schemes for Compliance in the Network

Though conformance and fairness to TCP is significant it however should not constrain the choices of end-to-end congestion control algorithms.The authors propose a class of non-linear TCP compatible congestion control schemes called Binomial Congestion Control Schemes (BCCS). AIMD, can be considered as one of congestion control schemes in the subset of TCP Compatible BCCS. Formally, the Binomial Congestion Control scheme can be defined as:

$$Wt+R \leftarrow Wt + \alpha / Wk \quad \text{if no loss} \tag{2.1}$$

$$Wt+\Delta t \leftarrow Wt - \beta Wt \quad \text{if loss} \tag{2.2}$$

where k and t are window scaling factors for increase and decrease respectively and α and β are increase the decrease proportionality constants. For any given values of α and β TCP Compatible BCCS can be defined by k+l = 1 : k>=0, l>=0.

Another interesting set of TCP Compatible congestion control algorithms has been presented in [7]. The proposal called Choose Your Response Function (or CYRF) has a general increase function f and a decrease function g which together constitute the congestion control policy. Formally the TCP-Compatibility is defined by the following constraints on the these two function f, g as:

$$f(x)g(x) \ \alpha \ x$$

There have also been other interesting proposals for TCP Compatible window based protocols [4] but covering all of them is beyond the scope of the thesis. Besides these window based proposals there have been suggestions for TCP compatible rate control scheme. The most popular rate based scheme is called TCP- Friendly Rate Control (or TFRC) . Since we have already discussed TFRC in Section 2.1.3 we do not elaborate on it any further.

2.4.2 Network Based Schemes for Compliance in the Network

Though there exists a range of end-system based TCP Compatible congestion control scheme they might still not meet the needs of various applications. Moreover there exists a possibility that end users may intentionally not use these algorithms. Therefore network based solutions are needed to enforce protocol compliance.

The network based support has been envisioned in two primary forms: a) schedulers and b) pricing mechanisms . Per-flow scheduling in form of Class Based Queueing, Priority Scheduling or Weighted Round Robin etc can be used to isolate flows, restrict bandwidth of misbehaving flows and thus provide TCP Compatibility. Similarly pricing mechanisms can also be used for differentiating against misbehaving flows by communicating them

higher price and thus ensuring TCP Compatibility in the network. However in order to achieve TCP Compatibility for the current Internet environment where flows compete in a FIFO queue all these mechanisms require tight coordination between all routers.

2.5 TCP – Friendly Marker Based Approach

For the last decade, researchers have studied and proposed improvements to TCP/ IP performance. These improvements include changes to TCP host implementations (e.g. Reno SACK etc.), better active queue management algorithms RED, FRED etc. or schemes which require both end-system and bottleneck support (e.g. ECN, adaptive packet marking). Packeteer's rate control is a buffer management scheme which manipulates TCP headers and acknowledgment (ack) rates to perform explicit, transparent control of TCP flows in certain niche deployment scenarios. Performance improvements of all these approaches have been measured in terms of timeout avoidance, better filtering of burst loss to determine congested periods and window reductions, optimization of retransmission, and improved fairness across competing TCP sessions (including both small, http-like and large, ftp-like sessions).

2.5.1 TCP-Friendly Best Effort Services

The term TCP-Friendly has been recently used in literature to characterize the behavior of non-TCP end-to-end congestion control schemes. The use of this term, in contrast, is to characterize building blocks that support superior best-effort performance for TCP applications. The key question then becomes: "What building blocks aspects matter for superior TCP performance?" It is well-known that timeouts are the leading source of TCP performance degradation from several perspectives such as average goodput, transfer-time and fairness. Therefore the goal of TCP-Friendly building blocks is to reduce the probability of timeouts. From an operational perspective TCP times out when:

- It loses all the packets or acks corresponding to a single window;
- Its window is very small (< 4*MSS) and it loses even one or two packets;
- Retransmitted packets are lost.
- It runs out of duplicate acks during a fast recovery phase;
- A burst of at least three closely spaced packets within a window are lost.

The last two conditions apply only to Reno, not SACK. Active queue management schemes, such as RED [10] and FRED , try to address some of these issues and hence they are "TCP-Friendly" in the sense of the definition. However, we note that the RED/ FRED approach is to randomize the dropping behavior and therefore they cannot provide a high level of assurance that timeouts would be contained, especially during heavy loads and/or during high degrees of TCP multiplexing. There are also end-system proposals to reduce the TCP dupack threshold and make other TCP level implementation improvements to address this issue. This proposal leverages the dimension of packet marking by extending the recent work in the following ways:

- Mark packets to protect small-window (< 4*MSS) flows from packet loss.
- Mark packets to maintain maximum spacing between packets marked as "OUT" (low priority), within limits of the total number of tokens, in order to disperse the effect of aggregate bursty loss;
- Mark packets to protect retransmitted packets from encountering loss;

Of these, the idea of protecting retransmissions from loss results in a large performance improvement because it affects all TCP implementations and accounts for a large fraction of retransmission timeouts. Retransmissions are generally sent during episodes of congestion and experience a higher loss probability. Moreover, in several cases, retransmission loss leads to a timeout in TCP Reno/ SACK and hence accounts for a disproportionate share of total number of timeouts. Therefore, protecting retransmissions

results in remarkable reductions in the number of timeouts, especially if the window sizes are small as well.

Now we consider the meaning of better best-effort service from a (TCP) performance and deployment perspective. Best-effort is the service commonly available on the Internet. Quantitative per-flow guarantees for best-effort performance are generally not given to customers. However, quantitative per-flow statistics over the population of users can be formulated for engineering purposes. In particular, user metrics (timeouts, average per-flow goodput, and coefficient of variation of per-flow goodputs) and network operator metrics (utilization, queue length, and packet loss rates) can be used to measure best-effort TCP/IP performance. These two perspectives are necessary because otherwise, one could optimize user-perceived performance with aggressive control schemes at the expense of network stability and other well-behaved flows. Similarly, one could optimize the network operator perceived performance while inflicting a large service variation on users. In this sense, better best-effort means a superior tradeoff between these multiple metrics, or an improvement in both user and operator metrics. In particular, it needs to eliminate a large fraction of total timeout instances seen by TCP.

Architecturally, better best-effort service requires the deployment of new components in the network, or the upgrade of end-systems. Here premise is that the incremental deployment of differential dropping and transport-aware marking components (e.g. through diffserv) can lead to such services. In this context, assuming that the network supports differential dropping mechanisms (e.g. RIO), this TCP-friendly marker will provide better best-effort or enhanced assured services for the subset of users who deploy these markers.

2.5.2 Performance Problems of TCP Over Assured Service

It is well known that TCP Reno (the large installed base of TCP implementations) has performance problems if a connection encounters a *burst loss of packets* i.e., if a connection sees a number of packet losses with nearby sequence numbers [9].

Specifically, three or more packets dropped in a window can lead to a *timeout* plus multiple SSTHRESH reductions. The distribution of these losses in the window is irrelevant for TCP Reno, i.e., the performance problems can occur with or without RED-type gateways as long as multiple losses are experienced by the same flow. Moreover, TCP transmission is bursty in the sense that blocks of packets are transmitted back-toback followed by idle times. Even after multiplexing at queuing points, packets of multiple flows generally tend to exhibit a low degree of interleaving. As a result when they encounter a bottleneck, multiple successive packets of a single flow have a high probability of experiencing similar behavior, for example, get dropped. Newer TCP Implementations like NewReno [7] and TCP SACK [6] address some of these problems by use of better end-to-end filtering, retransmission and feedback algorithms.

2.5.3 TCP-Friendly Packet Marker

A packet marker is one of the traffic conditioners in diffserv. The general problem in a marker is to optimally allocate an available pool of tokens to a set of incoming packets in a given interval of time. The available pool of tokens may depend upon service parameters such as the contracted rate and a measure of burstiness. Packets which get a token are said to be marked "IN" and those which do not get tokens are said to be marked "OUT." As mentioned earlier, this can be used as the basis of a simplified form of the "assured service" [1]. This algorithm can be generalized to the full assured service specification as well.

The Packet Marker is designed to :

1. **Protect small-window flows from packet losses:**

Small-window flows are "protected" from packet losses by allocating only "IN" tokens to them (subject to the availability of tokens). A "Max-Min"' fair [10] allocation of the rest of the tokens is made among the remaining flows. If there are not enough "IN" tokens to provide all small window flows with their demand, the number of available tokens is

equally divided among the small window flows and the remaining packets are marked as "OUT".

2. Maintain optimum spacing between "IN" and "OUT" tokens allocated for a flow:

There could be scenarios wherein a flow gets a burst of "OUT" tokens followed by a burst of "IN" tokens. This could lead to burst loss of "OUT" packets resulting in a timeout with TCP Reno. The probability of timeout could be reduced if an optimal spacing is maintained between "IN" and "OUT" packets for each flow. This allocation of tokens is done once every T seconds. This is illustrated in the Token Allocation algorithm

3. Mark packets according to the allocations:

Packet marking is done according to the per-flow allocations of tokens made in the previous two steps. In the marking algorithm, a packet is identified as belonging to a flow. All packets from small-window flows are marked as "IN". Also while marking, a spacing variable (number of consecutive "OUT" packets between every two "IN" packets of flow) is maintained on a per-flow basis. If "IN" tokens run out for any flow (either due to a prediction error or a sudden burst), then all successive packets for that flow are marked as "OUT". Figure 2.1 further illustrates the marking scheme. Note that in spite of the packet "interleaving" introduced by this marking scheme, there is a residual risk that a burst loss could result in multiple (though not consecutive) packet losses within a TCP window.

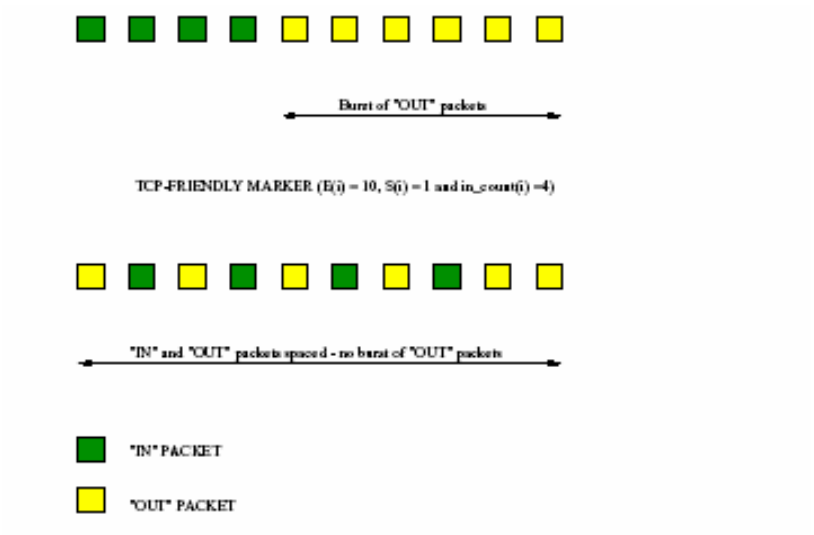

Figure 2.1: Illustration of marking scheme

2.6 The "Expected Capacity" Framework

2.6.1 Overview

The general approach of this mechanism is to define a *service profile* for each user, and to design a mechanism in the router that favors traffic that is within those service profiles. The core of the idea is very simple: monitor the traffic of each user as it enters the network, and tag packets as being "*In*" or "*Out*" of their service profiles. Then at each router, if congestion occurs, preferentially drop packets that are tagged as being "*Out*".

2.6.2 Location of Profile Meters in the Network

Figure 2.2 illustrates the "Expected Capacity" framework with a sender-based control. All the gateways (*G*) in the network have adopted a preferential dropping algorithm (*D*). In the simple sender based scheme, the function that checks whether traffic fits within a profile is implemented by tagging packets at the edge of the network, e.g., the profile meter (*M2*) is on the access link from H1 to ISP1. The complete story is more complex. A profile describes an expectation of service obtained by a customer from a provider. These relationships exist at many points in the network, ranging from individual users and their campus LANs to the peering relationships between global ISP's. Any such boundary may be an appropriate place for a profile meter, e.g., M3 to M6 in figure.

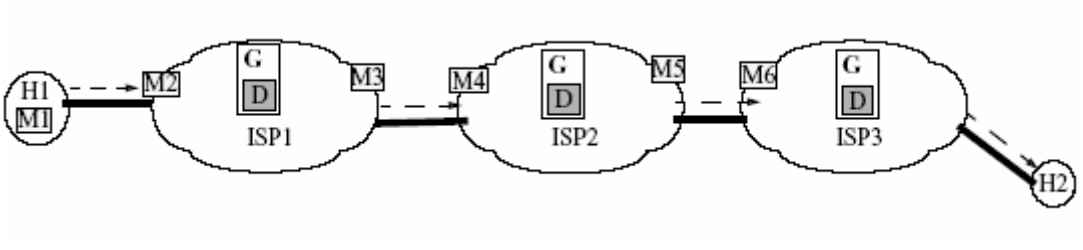

Figure 2.2 the "Expected Capacity" framework (sender-based)

Host 1, H1 has a sender-based profile, and is sending traffic to host H2 (dotted line). The traffic traverses three ISPs. The gateways, G's, in the figure are all augmented with preferential dropping algorithms, Ds. There are profile meters, Ms, at each interface between a customer and an ISP, or between two ISPs. M1 is a profile meter inside M3 M5 M6 M4 a host; M2 is on the access link from H1 to ISP1; M3-M6 are profile meters on the boundaries of ISPs.

Furthermore, the packet tagging associated with this service profile will, in the general case, be performed by devices at both side of a boundary. One such device, located on the sourcing traffic side of a network boundary, is a "policy meter" (M1, M3, and M5 in figure 2.2). This device chooses which packets to tag, based on some administrative policy. Another sort of device, the "checking meter", sits on the arriving traffic side of a network boundary, checks the incoming traffic, and marks packets as *Out* if the arriving

traffic exceeds the assigned profile, e.g., M2, M4 and M6. In this generalized model, a packet will travel through the network passing a series of cascaded profile meters.

2.6.3 A spectrum of services

The design of this framework serves two potentially conflicting goals. First, implements a set of simple services which are useful and easy to understand and adopt; second, it doesn't embed the above services into the mechanisms so that the framework cannot adapt to new applications with new service requirements in the future. The decoupling of the service profiles at the edge of the network from the differential dropping in the center of the network allows this flexibility. To over simply, the preferential dropping scheme adopted in routers in the center of the network will not change over time. Since the characteristics of a service is defined and captured by its corresponding profile meter, it is only necessary to create the profile meter at the edge of the network to adopt a new service.

2.6.4 Provisioning with Statistical Assurance

The statistical multiplexing nature of the Internet makes efficient use of bandwidth and supports an increasing number of users and new applications. However, it does lead to some uncertainty as to how much of the bandwidth is available at any instant. This approach to allocating traffic is to follow this philosophy to the degree that the user can tolerate the uncertainty. In other words, it believes that a capacity allocation scheme should provide a range of service assurance. At one extreme, the user may demand an absolute service assurance, even in the face of some network failures. Less demanding users may wish to purchase a service profile that is "usually available", but may still fail with low probability. The presumption is that a higher assurance service will cost substantially more to implement.

2.6.5 Receiver-Controlled Scheme

The tagging scheme described above implements a model in which the sender, by selecting one or another service profile, determines what service will govern each traffic flow. However, in today's Internet, the receiver of the traffic, not the sender, is often more the appropriate entity to make such decisions. A mechanism that implements receiver control of service, which is similar in approach and complementary to the sender controlled tagging scheme is described here.

The receiver-based scheme in the "Expected Capacity" framework is the dual of the sender-based scheme. It relies on a newly proposed change to TCP called the Explicit Congestion Notification (ECN) bit. In ECN semantics, congested gateways will turn on the ECN bit in a packet instead of dropping the packet. The TCP receiver copies the ECN bit into the acknowledgment (ack) packet, and the sender TCP will gracefully slow down upon receiving an ack with the ECN bit on.

2.6.5.1 Difference Between Sender-Based Control and Receiver-Based Control

There are a number of interesting asymmetries between the sender and the receiver versions of this tag and profile scheme, which arise from the fact that the data packets flow from the sender to the receiver. In the sender scheme, the packet first passes through the meter, where it is tagged, and then through any point of congestion. In contrast, in the receiver controlled scheme the packet first passes through any points of congestion, where it is tagged, and then through the receiver's meter. The receiver scheme, since routers only set the ECN bit if congestion is actually detected, can convey to the end point dynamic information about the current congestion levels. In the sender scheme, in contrast, profile meters must tag the packets as *In* or *Out* without knowing if congestion is actually present. Thus, the need is to construct a service, based on the receiver scheme, to bill user for actual usage during congestion.

On the other hand, the receiver scheme is more indirect in its ability to respond to congestion. Since a packet carries the explicit assertion of whether it is *In* or *Out* of profile in the sender scheme, the treatment of the packet is evident when it reaches a point of congestion. In the receiver scheme, the data packet itself carries no such profile indication, so at the point of congestion, the gateway must set the ECN bit, and still attempts to forward the packet, trusting the sender will correctly adjust its transmission rate. Of course, if the profile meter at the receiver's side employs a dropping algorithm, which will drop any packets that has exceeded the profile, the sender will slow down if it is a properly behaved TCP.

2.6.6. "Expected Capacity" for Bulk Data Transfers

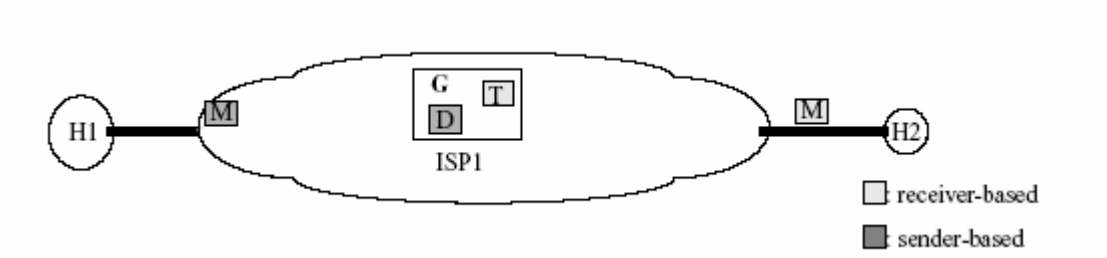

Figure 2.3 Simplified "Expected Capacity" framework

.

The simplified framework to illustrate the different levels of service provided for bulk-data TCP transfers. Both sender-based and receiver-based are shown. The darkly shaded boxes, profile meter M, and dropper D are for sender-based control. The lightly-shaded boxes, tagger T in the router and profile meter M at the receiver for receiver-based control. G are all the gateways in ISP1. There are no cascaded profile meters.

It is important to realize that the dropping algorithm in the routers, once adopted, is unlikely to change again over time; however, the service profiles and corresponding

profile meters will evolve as users have more sophisticated needs for new applications. Therefore, the need is to find a dropping algorithm that will offer enough generality to co-operate with many types of profile meters. In this section, we mention a preferential dropping algorithm to create discrimination in the center of the network. Additionally, we present a tagging algorithm tailored for bulk-data TCP transfers. For the sake of simplicity, we assume a simplified network with only one ISP between any two connecting hosts, as illustrated in figure 2.3. There is no cascading of profile meters. The service profiles used here are easy for the users to understand: they can provide a specific average throughput to anywhere within this network, with round trip times ranging from 20ms to 100ms (which is roughly comparable to metropolitan connections and cross-U.S. connections, respectively). We call the expected throughput the *target rate*, or RT. Different levels of service refer to the different target rates specified in service profiles.

2.6.7 TCP Rate Adjustment in the Current Internet

In today's Internet, rate adjustments are accomplished by both the end host transport layer TCP and the queue management algorithm in the gateways. We will discuss both in turn.

The mechanisms used by TCP to deal with congestion are based on [Jacobson]. TCP has two modes of dealing with congestion. The first mode, "Fast-Recovery", is triggered by the loss of very few packets, typically one. In this mode, the TCP cuts its sending window size in half, and following a successful retransmission, increases its window size by one packet each round trip time. Since the achieved transmission rate for any window size is roughly proportional to that window size, cutting the window size in half has the effect of reducing the achieved sending rate by some amount up to half. The second mode is called "Slow-Start", and typically occurs when a large number of packets are lost. Current implementations of TCP fail to recover multiple packet losses within a window, and have to rely on the retransmission timer to recover. When multiple packet losses occur, current TCP implementations usually remain silent until the retransmission timer goes off, and then enters "Slow-Start" mode. This has a more drastic effect on the TCP performance. First, the retransmission timer is crude, usually measured in a granularity of

500ms, and TCP does not send data during this period. Second, in Slow-Start, the sending TCP sets its window size to one packet when it starts again, with a much reduced Slow-Start threshold, *ssthresh*. Since *ssthresh* is what TCP perceives as the optimal operating point, and TCP opens up window linearly after its window size reaches *ssthresh*. This essentially reduces the sending rate to zero. Therefore, the rate adjustments currently implemented by TCP are both imprecise and, on occasion, drastic. Given this, there is a concern that TCP's rate adjustment mechanism cannot be used with enough precision to achieve a specific overall throughput, especially if Slow-Start is triggered.

2.7 Differential Dropping in the Routers: RIO

RIO stands for Random Early Drop (RED) gateways with *In/Out* bit. RED gateways [Floyd93] keep the overall throughput high while maintaining a small average queue length, and tolerate transient congestion without causing global synchronization. RIO retains all these attractive attributes. In addition, it discriminates against *Out* packets in times of congestion. At a high level, RIO uses twin RED algorithms for dropping packets, one for *In*s and one for *Out*s. By choosing the parameters for respective algorithms differently, RIO is able to discriminate against *Out* packets. We will briefly describe the RED algorithm before presenting RIO.

2.7.1 RED Algorithm

A RED gateway operates as follows: it computes the average queue size, and when the average queue size exceeds a certain threshold, it drops each arriving packet with a certain probability, where the exact probability is a function of the average queue size. The average queue size is calculated using a low-pass filter from instantaneous queue size, which allows transient bursts in the gateway. Persistent congestion in the gateway is reflected by a high average queue size and a high dropping probability. The resulting high dropping probability will discard packets early, detect and control congestion.

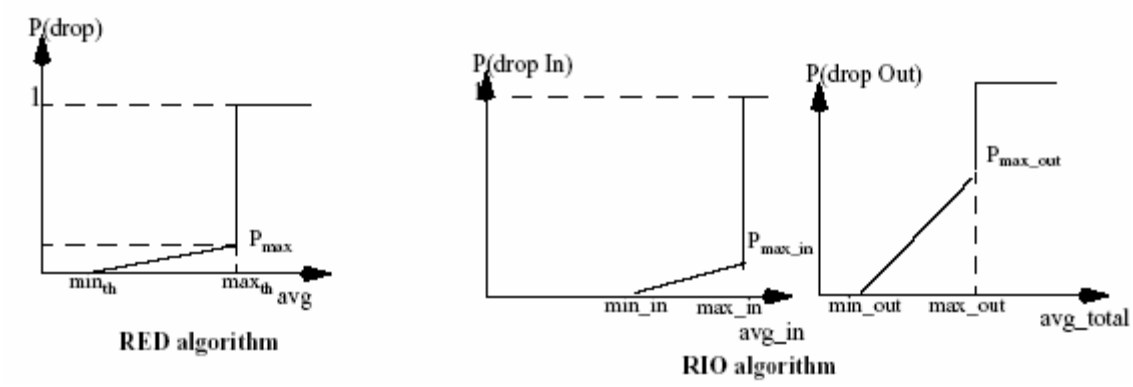

Figure 2.4 RED and RIO algorithms (figures are not drawn scale)

A RED gateway is configured with the following parameters: *minth*, *maxth*, and *Pmax*. It works as illustrated in the leftmost figure in figure 2.4: the *x* axis is *avg*, the average queue size, which is calculated using a low-pass filter of instantaneous queue size upon each packet arrival. The *y* axis is the probability of dropping an arriving packet. There are three phases in RED, defined by the average queue size in the range of [0, *minth*), [*minth*, *maxth*), and [*maxth*, ∞), respectively. The three phases are normal operation, congestion avoidance and congestion control, respectively. During the normal operation phase, when the average queue size is below *minth*, the gateway does not drop any packets. When the average queue size is between the two thresholds, the gateway is operating in the congestion avoidance phase, and each packet drop serves the purpose of notifying the end host transport layer to reduce its sending rate. Therefore, the dropping probability is a fraction of *Pmax*, and is usually small. When the average queue size is above *maxth*, which is usually caused by sustained large queue size, the gateway drops every arriving packet hoping to maintain a short queue size.

2.7.2 Twin Algorithms in RIO

RIO uses the same mechanism as in RED, but is configured with two sets of parameters, one for *In* packets, and one for *Out* packets. Upon each packet arrival at the gateway, the

gateway checks whether the packet is tagged as *In* or *Out*. If it is an *In* packet, the gateway calculates *avg_in*, the average queue for the *In* packets; the gateway calculates *avg_total*, average total queue size for *all (both In and Out)* arriving packets. The probability of dropping an *In* packet depends on *avg_in*, and the probability of dropping an *Out* packet depends on *avg_total*.

As illustrated in figure 2.4, there are three parameters for each of the twin algorithms. The three parameters, *min_in*, *max_in* and *Pmax_in*, define the normal operation [0, *min_in*), congestion avoidance [*min_in*, *max_in*), and congestion control [*max_in*,) phases for *In* packets. Similarly, *min_out*, *max_out*, and *Pmax_out* define the corresponding phases for *Out* packets.

The discrimination against *Out* packets in RIO is created by carefully choosing the parameters (*min_in, max_in, Pmax_in*), and (*min_out, max_out, Pmax_out*). As illustrated in two right figures in figure 2.4, a RIO gateway is more aggressive in dropping *Out* packets on three accounts: first, it drops *Out* packets much earlier than it drops *In* packets, this is done by choosing *min_out* smaller than *min_in*. Second, in the congestion avoidance phase, it drops *Out* packets with a higher probability, by setting *Pmax_out* higher than *Pmax_in*. Third, it goes into congestion control phase for the *Out* packets much earlier than for the *In* packets, by choosing *max_out* much smaller than *max_in*. In essence, RIO drops *Out* packets first when it detects incipient congestion, and drops all *Out* packets if the congestion persists. Only as a last resort, occurring when the gateway is flooded with *In* packets, it drops *In* packets in the hope of controlling congestion. In a well-provisioned network, this should never happen. When a gateway is consistently operating in a congestion control phase by dropping *In* packets, this is a clear indication that the network is under provisioned.

2.7.3 Profile Meters for Bulk Data Transfers: TSW Tagger

The profile meter designed for bulk-data transfers is called Time Sliding Window (TSW) tagger. TSW tagger has two distinct parts: a rate estimator and a tagging algorithm. TSW refers to the rate estimator algorithm. TSW provides a smooth estimate of the TCP sending rate1 over a period of time. With the estimated rate *avg_rate*, the tagging algorithm can tag packets as *Out* packets once the traffic exceeds a certain threshold.
TSW rate estimator works as follows:

Initially:

```
Win_length = a constant;
Avg_rate   = connection's target rate, R_T;
T_front    = 0;
```

Upon each packet arrival, TSW updates its state variables as follows:

```
Bytes_in_TSW = Avg_rate * Win_length;
New_bytes    = Bytes_in_TSW + pkt_size;
Avg_rate     = New_bytes / (now - T_front + Win_length);
T_front      = now;
```

Whereas, ***now*** is the time of current packet arrival; and ***pkt_size*** is the packet size of the arriving packet.

In terms of tagging algorithm, there are two different approaches. Ideally, a profile meter can keep a TCP connection oscillating between 0.66 RT, the target rate, and 1.33 RT so that, on average, the connection can achieve RT. The first approach is that the meter could remember a relatively long past history in the order of a TCP sawtooth from 0.66RT to 1.33RT and tag packets as *Out* with $P = (avgrate - RT) / (avgrate)$, when the *avg_rate* exceeds RT. All packets are tagged as *In* when the *avg_rate* is below RT. The second approach is for the profile meter to remember a relatively short history on the order of an RTT and look for the peak of a TCP sawtooth when TCP exceeds 1.33RT, at which point, the tagger starts tagging packets as *Out*. When the profile meters are next to the host, where TCP sawtooths are quite visible, the second approach is more effective.
On the other hand, the first approach is more general, and can be applied not only to individual TCP connections, but also aggregated TCP traffic or other type of traffic.

2.7.4 Difficulties in Designing RIO-TSW

The service profile: “certain target throughput to anywhere (within ISP)”, albeit simple, is in fact difficult to accomplish if the profile meters are on the access link from the hosts to their ISPs. There are two reasons for this. First of all, with the TCP algorithm for opening

up windows, there is a strong network bias in favor of connections with short round trip times (RTTs). In the Fast- Recovery phase, TCP increases *cwnd* by one packet each round trip time. Let *r* denote round trip time. Each round trip time, TCP increases its sending rate by 1/*r* pkts/sec, or it increases its sending rate by 1/(*r*)2 each second. For example, when a connection has an RTT five times that of another connection, the increase in sending rate for this connection is 1/25 of the other connection. Therefore, when both connections receive drops simultaneously, it takes the long RTT connection much longer to recover to its sending rate before the drop than the short one. During this period of recovery, the short RTT connection has a higher average sending rate than the long RTT connection. This explains why a service profile with specific source destination pair is comparatively easier to implement, because its RTT is known.

3. QUALITY OF SERVICE (QOS) MODELS

With the advent of diversified network applications and utilization, the services demanded by users demonstrate divergent requirements. The concept of QoS is introduced to meet the challenge of the problem. In recent years, various issues of QoS have been under intensive research and study in both the academic community and industry. In this chapter, we present a framework for the emerging QoS.

3.1 Absolute Differentiation vs. Relative Differentiation

In the traditional Internet, network nodes forward all the packets without discrimination. Packets are processed as quickly as possible. With the rapid transformation of the Internet into a commercial infrastructure, demand for service quality has rapidly developed. An adequate understanding of the different demands of service can serve as a foundation on which research and development work is conducted. In this section, we identify the measures of service quality and introduce two types of service differentiation.

Based on the two types of requirements, the differentiation of service quality can be categorized as absolute differentiation and relative differentiation. In the absolute

differentiation model, the network provides a set of service classes with different quantitative quality criteria, whereas in the relative differentiation model, the only promise from the network is that the quality of service received by a higher class user is better than that received by a lower class user.

3.2 Service Models

For the past few years, the architecture, requirements and mechanisms for providing service differentiation in the Internet have been the focus of extensive research. These research efforts have identified two fundamentally different approaches for service differentiation: Integrated Services (IntServ) and Differentiated Services (DiffServ).

3.2.1 The Integrated Services (IntServ) Model

The architecture of the Integrated Service consists of two elements: the IntServ model and the reference implementation framework. The IntServ model, which defines the externally visible behaviors, proposes two types of services targeted towards real-time traffic: the guaranteed service and the predictive or controlled load service.

Aiming at hard-real-time requirements, the Guaranteed Services provide firm bounds on queuing delay. The Controlled-Load Services are designed for adaptive real-time applications such as audio/video playback applications. Traffic belonging to the Guaranteed Services should expect bounded delay and zero loss whereas the Controlled-Load Services flows should expect to experience little queuing delay and little congestion loss.

The framework of IntServ consists of four components: the admission control routine,

the reservation setup protocol, the classifier and the packet schedule. Each component assumes the corresponding function in the setting up, forwarding and traffic control of a QoS packet flow.

The IntServ model is based on a solid background of research in QoS mechanisms and protocols. However, the acceptance of IntServ in industry is limited, at least so far. This is due to the scalability and manageability problems. Since IntServ focuses on individual packet flow, it requires nodes in the network core to keep the state of each flow. This is not viable for a network core node handling thousands of connections simultaneously. Although extensions to RSVP enable flow aggregation at the network core, the demanding requirement of IntServ on router, which requires all the routers along the path supporting RSVP, the admission control, the Multiple Field (MF) classification and the packet scheduling, makes it a impediment to the deployment of IntServ. In addition, end-to-end QoS support requires a multilateral service level agreement to be reached.

3.2.1.1 Guaranteed Services

Guaranteed service provides firm bounds on end-to-end queuing delays. It is designed for inelastic applications such as real-time applications. It thus guarantees both delay and bandwidth. Neither the setup mechanism nor the flow identification method is part of the guaranteed services model. Clearly all the network elements along the path of the network flow should achieve the bounded delay requirements.

3.2.1.2Controlled Services

Controlled Services is the solution for network elements that require multiple levels of qualities of service. This type of network behavior is required by tolerant and adaptive applications such as adaptive real-time applications. Applications requesting controlled-

load service provide the intermediate network elements with an estimation of the data traffic they will generate. Each network element accepting a request for controlled-load service must ensure that adequate bandwidth and packet processing resources are available to handle the requested level of traffic via active admission control. Packets belonging to flows that are not conforming to these estimates are either delayed or dropped.

3.2.2 The Differentiated Services (DiffServ) Model

DiffServ has been a focus of research within the QoS community in recent years. Due to the slow deployment of IntServ/RSVP, people have worked around the problem with a new service model.

The architecture of the DiffServ model is composed of a number of functional elements. It includes a small set of per-hop forwarding behaviors, packet classification functions, and traffic policing functions such as metering, marking and shaping. To achieve scalability, complex classification and traffic policing functions are only implemented at the network boundary nodes. By applying Per-Hop Behaviors (PHB) to aggregates of flows based on the Differentiated Services (DS) Field (TOS Field) in the IP header, the per-application or per-customer forwarding state need not be maintained at the network nodes any more. Admission control and resource provisioning can be either dynamic or static. A new optional element, the Bandwidth Broker (BB), is introduced to supervise the overall resource and admission management within a DiffServ domain.

A number of fundamental principles make DiffServ have many advantages over the IntServ model. First, DiffServ does not enforce the end-to-end service. All service level agreements are bilateral, through either manual negotiation or automatic reconciliation between BBs. Second, DiffServ reduces the demand for network core nodes by pushing as much of complicated function, such as traffic policing and packet classifying to the edge nodes. The core nodes only implement a small number of PHBs. Third, the service provisioning and traffic conditioning policies are sufficiently de-coupled from the forwarding behaviors within the network interior to permit the implementation of a wide variety of service behaviors. All these principles ensure the scalability, flexibility and manageability of the DiffServ model. So far two types of PHBs, Expedited Forwarding (EF) and Assured Forwarding (AF), have been made to the standard track.

The EF PHB is used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through DS domains. Such a service appears to the endpoints as a point-to-point connection or a "virtual leased line". To achieve this service, two parts are needed: EF PHB and traffic conditioning at the edge nodes. EF PHB ensures the EF traffic aggregate has a well-defined minimum departure rate. Traffic conditioning, including policing and shaping, makes arrival rate at any node be always less than that node's configured minimum departure rate.

The AF PHB provides different drop precedence at the time of congestion. A typical service built upon AF PHB is called Assured Service, in which each user subscribes to a committed traffic profile. When a user's flow rate exceeds the profile, some of the

packets, which are marked as the out profile packets, are subject to a higher drop probability. In the DiffServ framework, the traditional best-effort traffic has the lowest forwarding probability.

Differentiated services are intended to provide a framework and building blocks to enable deployment of scalable service discrimination in the Internet. It avoids per-flow state and signaling at every hop by applying per hop behavior to aggregated traffic at network boundaries. The Per-Hop Behaviors (PHB) are defined to permit a granular way of allocating buffer and bandwidth resources within the network nodes along the path.

It uses the 8-bit IPv4 type of service (TOS) [8] header field to provide the information necessary for obtaining a particular level of service defined in the service level agreement (SLA). The TOS field is divided into the 6-bit Differentiated Services Code Point (DSCP) and 2-bit Currently Unused (CU). The DSCP field contains the information that informs the nodes along the path of the Per Hop Behavior (PHB) desired.

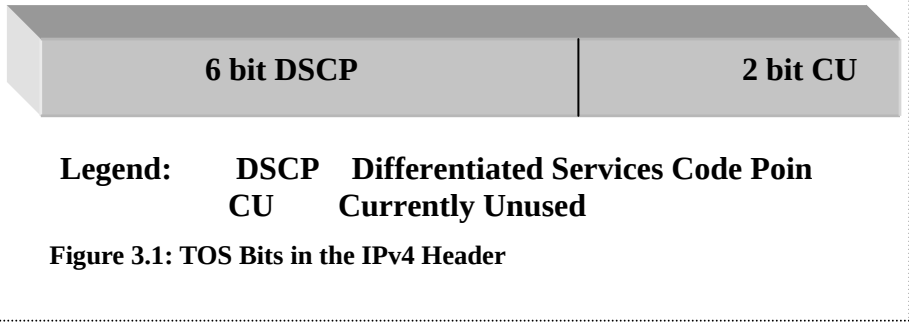

Figure 3.1: TOS Bits in the IPv4 Header

The DS nodes collectively form a DS domain and operate under a common service provisioning policy with a defined set of PHBs. DS boundary nodes classify and condition ingress traffic to ensure that packets that transit the domain are marked to select a PHB. Within the DS domain the forwarding behavior is selected based on the DSCP that maps to a PHB. DS domains establish service level agreements (SLA) with neighboring DS domains. The administration of the domain is responsible for ensuring

that the SLA is not violated. The SLA specifies the packet classification and re-marking rules. It also specifies the traffic profiles and actions applicable to the respective traffic streams.

The DS boundary nodes act as an ingress node and egress node for network traffic. Traffic enters a domain at an ingress node and leaves at an egress node. The ingress and egress nodes are responsible for ensuring that the traffic entering the DS domain conforms to the SLA established between their domain and the neighboring domain. The traffic conditioning agreement (TCA) that is derived from the SLA ensures that the respective nodes take appropriate action against non-conforming traffic. The DS nodes contain various functional elements for applying the TCA. These elements do forwarding, classification and traffic conditioning including metering, marking, shaping and policing.

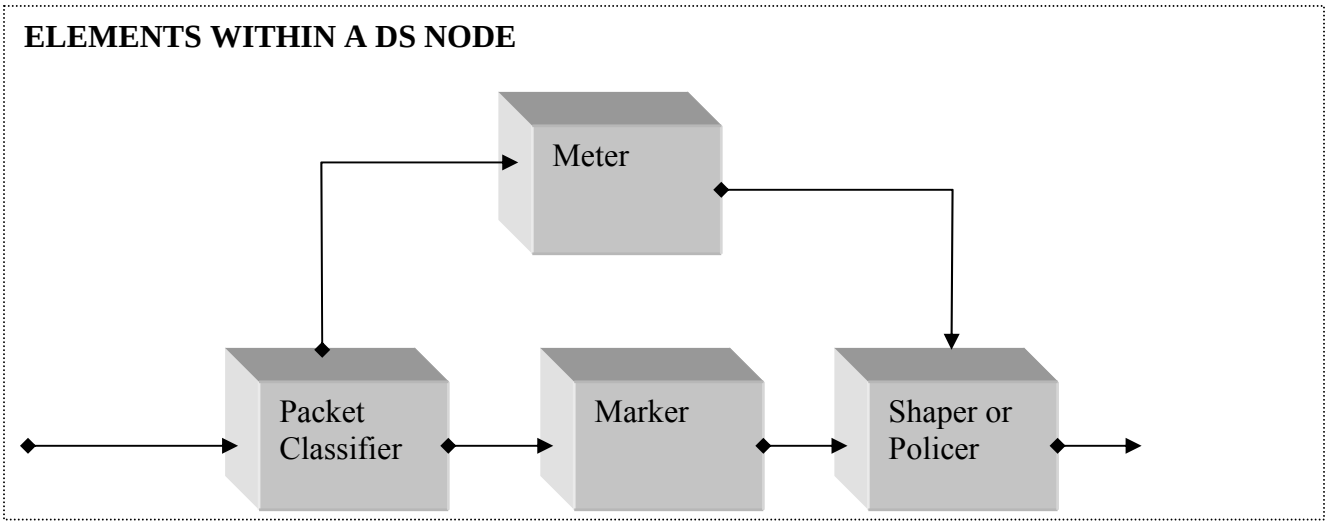

Figure 3.2: The Traffic Conditioning Functional Elements Within A Network Node

The packet classifier selects the packets in a traffic stream based on the content of some portion of the packet header. The BA (Behavior Aggregate) classifier classifies packets based on the DS codepoint only. The MF (Multi-Field) classifier classifies the packets based on the value of a combination of one or more header fields such as the source address, the destination address, the DS field, the protocol ID, the source port, destination

port numbers and incoming interface. The classifier then forwards the packets to the appropriate functional element for the flow to receive its assigned PHB.
The meter is used to measure the traffic stream against a traffic profile. The traffic profile specifies the traffic properties of a traffic stream selected by a classifier. It provides rules for determining whether a particular packet is in-profile or out-of-profile. These rules are specified in the TCA. These rules maybe applied using the Token Bucket algorithm or the Sliding Window algorithm. The in-profile packets are allowed to enter the DS domain without further conditioning. Out-of- profile packets may be queued until they are in-profile (shaped), discarded (policed), marked with a new codepoint (re-marked), or forwarded unchanged while triggering some accounting procedure.

A Per-Hop Behavior (PHB) is a description of the forwarding behavior of a DS node. The forwarding behavior, which includes loss, delay and jitter, normally depends on the relative loading of the link. A PHB is the means by which a node allocates resources to the behavior aggregates. A PHB can be specified in terms of the available bandwidth of a link, buffer space at a node or traffic characteristics such loss or delay.

A PHB is selected at a node by mapping the DS codepoint in a received packet [8]. The codepoints are assigned to the respective PHB simultaneously ensuring that IP precedence is maintained. Some codepoints can map to the same PHB but every codepoint must be mapped to some PHB. DS nodes must select PHBs by matching against the entire 6-bit DSCP field in the TOS field. Currently three broad levels of service have been defined. Expedited Forwarding (EF), Assured Forwarding and Best Effort (BE). The BE PHB is the default service provided. The BE PHB has a codepoint of 000000.

3.2.2.1Expedited Forwarding PHB

The EF PHB is for building a low loss, low latency and low jitter end-to-end service through the DS domains . It aims at simulating a Virtual Leased Line (VLL). Loss, latency and jitter are due to the queue traffic. To achieve this service provisioning is

required to ensure that the aggregate sees small queues. The arrival rate should be less than the departure rate and the minimum departure rate must be pre-configured.

Within the EF PHB framework, all non-conforming traffic violating the SLA is discarded. The EF PHB also ensures that the amount of traffic in the node should be provisioned in such a way so as to prevent starving other flows. The codepoint 101110 is recommended for the EF PHB.

3.2.2.1 Assured Forwarding (AF) PHB

The AF PHB [31] group is a means for a provider DS domain to offer different levels of forwarding assurances for IP packets received from a customer DS domain. The AF PHB group provides delivery of IP packets in four independently forwarded AF classes. Each AF class is allocated a certain amount of resources (buffer space and bandwidth). Within each AF class, an IP packet can be assigned one of three different levels of drop precedence. In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF class. A congested DS node tries to protect packets with a lower drop precedence value from being lost by preferably discarding packets with a higher drop precedence value. A DS node does not reorder IP packets of the same micro-flow if they belong to the same AF class. Figure 3.4 shows the respective arrangement.

AF CODEPOINTS

	Class 1	**Class 2**	**Class 3**	**Class 4**
Low Drop Prec	001010	010010	011010	100010
Medium Drop Prec	001100	010100	011100	100100
High Drop Prec	001110	010110	011110	100110

Figure 3.4 Codepoint Assignment For The AF PHB

The DSCP field within the DS field is capable of conveying 64 distinct codepoints. The

codepoint space is divided into three pools for the purpose of codepoint assignment and management. A pool of 32 codepoints (Pool 1) to be assigned for the standards action use. This pool is defined as xxxxx0 and is available for use. A pool of 16 codepoints (Pool 2) to be reserved for experimental or local use. This pool is defined as xxxx11. A pool of 16 codepoints (Pool 3) which are initially available for experimental or local use, but which should be preferentially utilized for standardized assignments if Pool 1 is ever exhausted. This pool is defined as xxxx01. Within pool 1, bit 5 and bit 6 is used for defining the drop precedence within each AF class. Bit 1, 2 and 3 are used for defining the AF class. Currently four classes are defined in the differentiated working group committee as shown in the figure 3.4.

3.2.3 The Rate Proportional Differentiation (RPD) Model

The work in this thesis is mainly motivated by the Rate Proportional Differentiation Model. Based on the observation that most current multimedia applications are capable of being adaptive to network conditions, that is, applications are able to tolerate a certain degree of latency, jitter and packet loss, the model proposes a network to provide a set of service classes. In particular, the differentiation among service classes takes the following form:

$$\frac{l_i(t,t+\tau)}{l_j(t,t+\tau)} = \frac{\phi_i}{\phi_j} \quad (1.1)$$

where ϕ_i is the Differentiation Parameters, and $l_i(t,t+\tau)$ can be either queuing delay or the inverse of the drop rate of the arrivals of class i during the interval $(t, t+\tau)$.

In addition, at the network edge, instead of applying traffic policing, traffic accounting is used. Users are charged on the service class usage. In this model, applications are considered as intelligent and adaptive; they use the lower service class as much as

possible and only migrate to a higher service class when the current service class cannot meet the quality of service demand.

3.3 Taxonomy on Service Models

So far, we have presented three service models. Table 3.1 shows the taxonomy on these service models based on the analysis in the first section of this chapter.

Table 3.1 The Taxonomy on Service Models

	Absolute Differentiation	Relative Differentiation
Integrated Services	Guaranteed Services	Controlled-Load Services
Differentiated Services	Premium Services	Assured Services
Rate Proportional Differentiation Model		Rate Proportional Loss Rate Proportional Delay

As shown in Table 3.1, both the Controlled-Load Service in IntServ and the Assured Service in DiffServ are categorized as providing the relative differentiation. The implementation detail and the mechanisms of these two services are different since they belong to different service differentiation models. However, from the perspective of external behavior, these two services are quite similar. They both need the user to subscribe to the service with the token bucket parameters specifying the required quality. The quality assurance of the service relies on the user's commitment to the subscribed traffic profile. In addition, probabilistic approaches are used in the admission control of both the service models. This implies that, although at low probability, under-provisioning can happen in the network and the users might receive a lower quality of services than the subscribed quality.

3.4 Other QoS related issues

Besides research on different service models, a number of related components have been under study within the context of QoS. In this section, we briefly discuss the basic concept of traffic engineering and traffic modeling.

3.4.1 Traffic Engineering and Constraint Based Routing

In a network, one reason for congestion is unevenly distributed traffic where some parts of the network are overloaded while other parts are lightly loaded. This is because the traditional route calculation is only based on relatively static information - the number of hops and the link delay. Traffic Engineering refers the process of arranging how traffic flows through the network so that congestion caused by uneven network utilization can be avoided. Constraint Based Routing, which evolves from the QoS routing, is an important mechanism for making the Traffic Engineering process automatic. In the OSI reference model, Constraint Based Routing resides in the third layer. The goal of Constraint Based Routing is:

1. To select routes that can meet certain QoS requirement;
2. To increase the utilization of the network **Error! Reference source not found.**.

Constraint Based Routing encompasses a number of aspects: route information distribution, QoS route computation, route establishment and route maintenance. For each aspect, there exist different approaches and algorithms. For example, for route information distribution, we can have either periodic updates or triggered distributions.

With regard to the timing of route computation, we can use either on-demand computation or pre-computation.

3.4.2 Traffic Modeling

The characteristics of data traffic play a crucial role in the performance analysis and the design of communication networks. Understanding the models of network traffic can help us design better protocols, better network topologies, etc. **Error! Reference source not found.**. In addition, within the context of differentiated service, we believe it also plays an important role for network service providers in increasing their revenue by setting up appropriate billing tiers. Traffic modeling provides a way to catch those traffic characteristics.

3.5 Summary

In this section, we presented an overview of the QoS service and a number of important components: the service models, traffic engineering and traffic modeling. This outlined a framework in which our research on the intelligent marker is conducted.

4. INTELLIGENT MARKER

In this section we describe the major design issues that were of concern for us and the algorithm that we propose.

4.1 Design Issues

TCP performance is highly influenced by two parameters, namely RTT, and window size. Hence, one of the challenges was to design a marker which understands the TCP dynamics and which helps in reducing the influence of RTT and window size on the performance achieved by the TCP flows. Since markers are mostly deployed at the edge routers, which cannot easily decide the window size and RTT of the various TCP connections passing through, our effort was to have a marker, which can indirectly sense the changes in these parameters and mark accordingly.

Another issue was to develop a marker, which is least sensitive to its own parameters unlike the existing markers mentioned in Sections 2. For example, TB–TCs are very much sensitive to the bucket parameters and the TSW–TCs are very much sensitive to the time window (i.e., the past history that the marker remembers).Still another concern was to reduce the burstiness of the marked and unmarked packets, to avoid the potential instability problem reported in [7].Our marking algorithm details clearly show how the first two issues are dealt with. The burstiness problem is resolved by means of

probabilistic marking while each flow (or aggregate) is both in-profile and out of profile, and also by adaptively changing these marking probabilities. Some of the other issues of importance were to have a simple algorithm which requires no support from the end hosts and hence be transparent to the end hosts, and to see that marking is optimal in the sense that while maintaining the observed rate close to the target rate, it should not mark more packets than required. That is, the assured service classes should obtain their fair share of the best effort bandwidth.

4.2 The Marking Algorithm

Taking the above issues into consideration, we came up with the algorithm for intelligent marker .As mentioned earlier; it is a TSW–TC and hence has the rate estimator which calculates the average rate as in [8] and the marker, which marks the packets, based on this average rate. The Intelligent marking algorithm is described as follows:

For each packet arrival

```
    if avg_rate <= cir
     then
        mp = mp + (1 – avg_rate/ cir) +  (par- avg_rate)/ avg rate;
        par = avg rate;
```

mark the packet using:

cp 11 w.p. mp
cp 00 w.p. (1 - mp)

else if avg_rate > cir

then

mp = mp +(par - avg rate) / avg_rate;

par = avg_rate;

mark the packet using:

cp 11 w.p. mp

cp 00 w.p. (1 - mp)

where avg_rate is the rate estimate upon each packet arrival; mp, the marking probability (<=1); cir, the committed information rate (i.e., the target rate); par, the previous average rate; cp denotes 'codepoint' and w.p. denotes 'with probability'.

Next we discuss the basis of our algorithm .The TCP window size W and the RTT are related to the throughput by the equation [1].

BW = ¾ (MSS * W) / (RTT)

where W is expressed in number of segments.

Any variation in W or RTT is reflected as subsequent changes in BW, i.e., in our case, the avg_rate. This is our basis of introducing the parameter previous average rate (par), which is compared with the present average rate to track any change in the rate of flow and thus indirectly extract the variations in RTT or W. We call this the Intelligent marking approach, because the par is used to take into consideration any instantaneous change in the average rate of the flow. During the period when TCP flows experience congestion, either or both of the following occur:

(a) The cwnd reduces reducing the value of W;

(b) The RTT increases.

In the expression for the marking probability mp, (par – avg_rate) / (avg_rate) tracks the variations in the above factors and thus increases or decreases the marking probability according to the changes in the flow rate, whereas (1 - avg_rate) / cir constantly compares the average rate observed with the target rate to keep the rate closer to the target. Thus, when the avg_rate is below cir but increasing, the factor (1 – avg_rate=cir) tries to increase the marking probability to reach the target, whereas the factor (par - avg_rate) / avg_rate tries to reduce the marking probability though at a lower rate. When the avg_rate is below cir, and still falling down, both the factors increase the marking probability. Similarly, it takes care of the instantaneous changes in the flow rate while avg_rate is above cir. This behavior of the marker plays a major role in improving the performance of TCP .We refer to packets with codepoint 11 as marked packets and those with codepoint 00 as unmarked packets in later sections of this thesis.

5. IMPLEMENTATION DETAILS

The studies in this thesis were performed using NS simulator on Red hat Linux 7.0. We used Nortel's DiffServ module for implementing it in NS, which we modified to incorporate our marking algorithm.

5.1 The Scenario

In this section we outline the topology and basic assumptions used for all our experiments described here. We consider a scenario where traffic flows between two corporate networks (CNs) via an ISP network, which is DiffServ enabled.

Table5.1
Simulation parameters

TCP segment size	536 bytes		
RTT	100 ms		
Simulation time	210 s		
TSW window length	1 s		
	Min_th (packets)	Max_th (packets)	Max_dp
Marked	250	500	0.02
Unmarked	150	300	0.1

We assume that all the intermediate routers have RIO based AQM mechanism. The RIO parameters and buffer size are suitably set in order to avoid any kind of bottleneck. The typical values used to get the results reported here are shown in Table 5.1. The topology is as shown in figure 5.1. All links from R1 to R5 are of same bandwidth, which is mentioned later with the respective experiments. The marker is placed only at the egress edge router R1 to CN1. S1 to Sn represent the sources and D1 to Dn represent the receiver for the experiment. R2 and R4 are the edge routers and R3 is the core router of the DiffServ domain.

5.2 Simulation Parameters

We used FTP bulk data transfer for the TCP traffic in all our experiments. Table 5.1 shows the values of common simulation parameters for all the experiments. Any deviation from the values specified in Table 5.1 would be mentioned in the respective experiments.

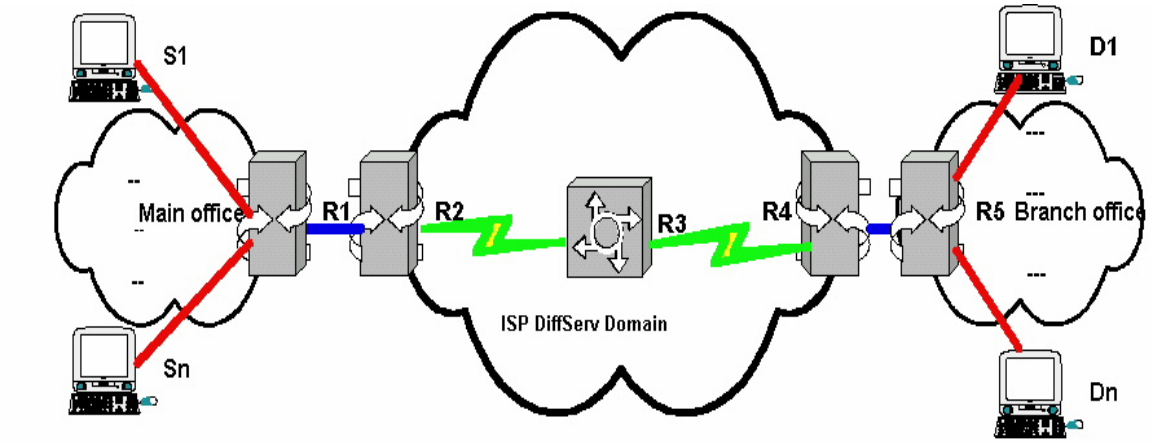

Figure 5.1 The topology

5.3 Results and Analysis

We conducted a series of experiments to analyze the effectiveness of our marker. It is to be noted that for all our experiments, we have measured the goodput, whereas the rate estimator calculates the sending rate as the avg_rate. We account this as the possible reason for some of the achieved rates being slightly less than the assured rate.

5.3.1 Assured Service for Aggregates with Different Target Rates

We did a set of experiments with different combinations of target rates to analyze the behavior of marker in the cases of under-, over-, and well-subscribed networks. The aim of these experiments was to study the capability of the marker to assure the target rate for

priority (AS) flows. We had two sets of priority TCP flows (each having six micro-flows), with aggregate target requirements, along with a set of nine best effort (BE) TCP micro- flows. The bandwidth of all the links were set to 10 Mbps. Table 5.2 summarizes the results obtained for various combinations of the target rates. The target rates of the two AS aggregates are indicated in columns 2 and 3.Next four columns show the achieved rates for these two aggregates. In addition to the total rates, we also show the component due to the marked packets in order to verify that the marking is optimal and the excess, i.e., best effort bandwidth, is equally shared among all the flows.

5.3.1.1Analysis

The results clearly show that the flows achieve the target rates in the under- and well-subscribed cases quite convincingly, and reach quite close to the targets in the over-subscribed case. As mentioned before, it is to be noted that we are measuring the goodput at the receiver whereas the marker uses the sending rate estimated by the TSW rate estimator. The results show that in the under-subscribed scenario, all the flows share approximately equal amount of the excess bandwidth. But in the over-subscribed regions, we see the priority flows getting a lesser share of the best effort. This is due to the fact that as the target requirement increases we see an increase in the marked packet rate in order to reach the target rates, which leaves very less amount of the unmarked packets for the AS TCP flows.

Table 5.2
Achieved rates (Ra) for different target rates (Rt)

Exp# 1	Target Rates (Mbps)		Achieved Rates (Mbps)				BE Tcp Flow (Mbps)	Link Goodput (Mbps)
	Rt 1	Rt 2	Ra 1		Ra 2			
			Total	Marked	Total	Marked		
1	1	1	2.85	1.45	3.35	1.97	2.94	9.14
2	1	2	2.93	1.76	3.6	2.7	2.64	9.17
3	1	3	2.93	2.08	4.08	3.44	2.2	9.21
4	1	4	2.93	2.21	4.29	3.84	1.93	9.15
5	1	5	2.8	2.32	4.89	4.64	1.51	9.2
6	2	2	3.4	2.58	3.56	2.73	2.49	9.45
7	3	3	3.75	3.34	3.53	3.08	1.85	9.13
8	4	4	3.88	3.7	3.94	3.7	1.31	9.13
9	5	5	4.38	4.38	4.35	4.35	0.42	9.15
10	6	6	4.35	4.35	4.5	4.5	0.34	9.19
Average link utilization = 92% (approx.)								9.192

5.3.2 Effect of Different RTTs

We next studied the effect of different RTTs on MBM.TCP shows an unfair bias against long RTT flows during congestion .Our aim in this experiment was to see if the MBM helps in reducing this bias. The experiment was performed with five pairs of flow aggregates, with different RTTs ranging from 60 to 140 ms.Each flow aggregate had six micro-flows in it. The link bandwidths from R1 to R5 (as shown in Fig. 5.1) were all set to 28 Mbps.The two aggregates of each pair had distinct target requirements of 1 and 4 Mbps and all flows in a pair had the same RTT.We set appropriate window sizes to avoid any bottlenecks due to it. We summarize the results in Table 5.3.

Table 5.3
Achieved rates (Ra) for different RTT values

RTT (ms)	Achieved rates (Mbps)		Per source pair
	Ra 1	Ra 2	goodput (Mbps)
60	1.82	3.81	5.63
80	1.49	3.74	5.23
100	1.52	3.52	5.04
120	1.38	3.58	4.96
140	1.43	3.45	4.88
Total link goodput			25.74

5.3.2.1 Analysis

From the above results, it is evident that MBM does manage to reduce the TCP bias against long RTTs.The difference in goodputs achieved by the low latency flow (60 ms

RTT) and the long latency flow (140 ms RTT) is only 0.75 Mbps or 13%. The flows with a target rate 1 Mbps achieve their targets and are unaffected by this difference. The overall link utilization in this case is 92%.

5.3.3 Effect of different window sizes

Different users may use different TCP implementations, which have different advertised window sizes by default. Next, we studied the behavior of MBM to TCP flows with different advertised window sizes.TCP is known to perform poorly if the window is not set to a value equivalent to the bandwidth–delay product .The objective of this experiment was to see the effectiveness of MBM in providing the assurance to the priority TCP flows with different window sizes, ranging from a low value to a higher than the bandwidth–delay product. The set-up had five assured TCP flows having the same RTT (500 ms) but different window sizes ranging from 384 to 1920 KB.The flows had a target rate of 3 Mbps. The link bandwidth from R1 to R5 (as shown in Fig.1) was all set to 18 Mbps.We ran experiments with and without MBM (using the same set-up) to compare the performance. The optimum window size for an RTT ¼ 500 ms and link bandwidth ¼18 Mbps is 1125 KB.The results of this experiment are summarized in Table 5.4.

Table 5.4

Achieved rates (Ra) for different window sizes

Window size (KB)	Achieved rates (Mbps)	
	w/o marker	with Marker
384	0.58	1.88
768	3.1	3.06
1125	3.21	2.87
1536	2.76	3.07
1920	1.25	2.93
Total link utilization	11.90	13.81

5.3.3.1 Analysis

Based on the results achieved in Table 5.4, we note that without MBM, the flows with window values closer to the optimum value receives a greater share of the link bandwidth, whereas the flows with lower window values suffer. However the goodputs achieved using MBM shows that the flows with a lower window (384 KB) gets a better share of the total bandwidth compared to the situation when there was no MBM.The overall link utilization with MBM (76.7%) is also higher than without MBM (60.5%). We believe that using TCP extensions-like SACK would help in achieving even better results with MBM.

6. INFERENCE AND FUTURE WORK

There is a growing need for intelligent TCP friendly markers in present day Internet. In this thesis, we presented an intelligent marker based approach in providing better quality of service especially for TCP flows. It stands out from other markers in its transparency from the end hosts, simplicity, and least sensitivity to parameters of both TCP as well as its own parameters. These claims have been substantiated in our experiments, which show that our markers help in achieving the target rate, with a better fairness in terms of sharing the excess bandwidth among flows. It also provides the TCP flows, a greater degree of insulation from differences in RTT and window sizes, which is one of the major causes of worry today. The overall link utilization also seems to be much better. This approach plays a major part in establishing these results as has been explained in the previous sections. In our experiments, we used New Reno TCP implementation. We believe that by using the TCP extensions such as SACK, our marker would provide even better results .Future work would include extending the present algorithm of the markers to take into consideration the congestion in the network based on a feedback architecture.

References:

[1] D. Lin and R. Morris," Dynamics of Random Early Detection," Proceedings of SIGCOMM'97, September 1997.

[2] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow," TCP Selective Acknowledgment Options", IETF Internet RFC 2018,October 1996.

[3] S. Floyd, and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, August 1993, pp.397-413.

[4] F.Azeem, A.Rao, X.Lu, S.Kalyanara man, TCP-friendly traffic conditioners for differentiated services, IETF Internet Draft, March 1999.

[5] W.Feng, D. Kandlur, D.Saha, K.Shin, Adaptive packet marking for providing differentiated services in the Interent, IEEE/ACM Transactions on Networking 7 (5) (1999) 685–697.

[6] W.Lin, R.Zheng, J.Hou, How to make assured services more assured, in: Proceedings of the 7th International Conference on Network Protocols (ICNP'99), Toronto, Canada, October 1999, pp.182–191 .

[7] D.D. Clark andW. Fang, "Explicit allocation of besteffort packet delivery service", *IEEE/ACM Transactions on Networking*, 6(4):362-373, Aug. 1998.

[8] S. Blake, D.Black, M.Carlson, E.Davies, Z.Wang andW.Weiss, "An architecture for differentiated services", *Internet RFC 2475*, December 1998.

[9] Das, Dutta, Helmy ," Fair stateless Aggregate traffic marking using active queue management techniques.

[10] Monaco, Feroz, Xia, Kalyanaraman, " TCP friendly marking for scalable Best effort services on the internet.