2004

Active congestion control using ABCD (available bandwidth-based congestion detection).

Aniruddha Bharadwaj
University of Windsor

Active Congestion Control Using ABCD
(Available Bandwidth-based Congestion Detection)

By

Aniruddha Bharadwaj

A Thesis
Submitted to the Faculty of Graduate Studies and Research
Through the School of Computer Science
In Partial Fulfillment of the Requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario
Canada
2004

Your file Votre référence
ISBN: 0-612-96376-4
Our file Notre référence
ISBN: 0-612-96376-4

Abstract

With the growth of the Internet, the problem of congestion has attained the distinction of being a perennial problem. The Internet community has been trying several approaches for improved congestion control techniques. The end-to-end approach is considered to be the most robust one and it has served quite well until recently, when researchers started to explore the information available at the intermediate node level. This approach triggered a new field called Active Networks where intermediate nodes have a much larger role to play than that of the naïve nodes. This thesis proposes an active congestion control (ACC) scheme based on Available Bandwidth-based Congestion Detection (ABCD), which regulates the traffic according to network conditions. Dynamic changes in the available bandwidth can trigger re-negotiation of flow rate. We have introduced packet size adjustment at the intermediate router in addition to rate control at sender node, scaled according to the available bandwidth, which is estimated using three packet probes. To verify the improved scheme, we have extended Ted Faber's ACC work in NS-2 simulator. With this simulator we verify ACC-ABCD's gains such as a marginal improvement in average TCP throughput at each endpoint, fewer packet drops and improved fairness index. Our tests on NS-2 prove that the ACC-ABCD technique yields better results as compared to TCP congestion control with or without the cross traffic.

Keywords – Active Networks, Congestion Control, Congestion Avoidance, Available Bandwidth

Dedicated to

'Aai - Aba' and my aspirations for ***'Doctor'...***

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

First and foremost I thank my thesis advisor Dr. A. K. Aggarwal for his patience and support throughput my graduate studies. His overly enthusiasm, deep insight and mastery of the subject matter have made working on this thesis a wonderful experience. I have had the opportunity to broaden and improve my analytical skills, as well as gain further understanding of how to formulate and solve intricate problems. I am deeply indebted to him for keeping faith in my abilities

I owe lots of gratitude to my thesis committee Dr. Kai Hildebrandt, Dr. A. Jaekel and Dr. S. Bandyopadhyay, who kept an eye on the progress of my work and for their constructive criticism and valuable suggestions.

Special thanks to Dr. Subir & Mr. Adlane Habed, working with them for more than two years was like an honor. I would owe my career in academia to these people.

I would also like to thank Dr. Frost, Dr. Kent, Dr. El-Marakby and Prof. Jackson Carvalho for their memorable graduate courses.

I gratefully acknowledge financial support provided by the Graduate Studies (University of Windsor) and the research facilities provided by the School of Computer Science.

I appreciate kind help of the administrative staff (Mary, Margaret, Gloria, Debbie and Roxana) and support staff (Walid, Anas, Maunzer, Brian and Sanjay), who were very helpful despite me being relentlessly demanding. Tireless attitude of these people was a motivation itself.

A special mention of all the colleagues (the batch of winter 2002) and the fellow research group members of HPC group for their support and encouragement. I have to thank all of you as a unit because we always worked as a group. Keep up this spirit.

Interdependence is certainly more valuable than independence. This thesis as well as all the other little efforts have been accompanied and supported by my family members (15). I am fortunate to be one of them. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

I am thankful to my grandfather (Nana) as well as both the grandparents, some of them I haven't seen and some I don't remember well. But their family values and qualities have helped me to form my vision.

Words are not enough to express gratitude towards my parents (Aai - Aba). They always believed in me even when, I lost the self-confidence. Thank you very much for inspiring and guiding this humble being.

Many thanks to Baba-Kaku, Anna-Kaku, Dada-Wahini and Pada-Shilpa, these are the people without whom I would not have been able to get such a quality education. I am thankful to them for their support and understanding.

Earnest thank to Akka-Bhauji for being my guardians or I can say for being my parents in North America. I am thankful to my sisters Didi, Guddi, Pinki ani Sonu for rendering me the sense and the value of sisterhood.

Again thanks to Pada-Shilpa for everything they have done for me. And lastly, thanks to generation next of our family Prathamesh, Gargee, Pranali, Prachiti, Saket and many more to come...

The chain of my gratitude would be definitely incomplete if I would forget to thank the first cause of this chain, using Aristotle's words, The Prime Mover. My deepest

and sincere gratitude towards Lord Ganesh for bestowing upon me, the kind blessings, that gave me the confidence to fulfill expectations of my family.

Above all I owe my career to my failures, which taught me how to fight out my way.

Aai & Aba, I am not able to fulfill your all dreams but truly whatever I have achieved, is just for you.

Aniruddha Bharadwaj
'Sankashti Chaturthi' (September 2, 2004)
Windsor.

Table of Contents

List of Figures

List of Tables

Chapter 1 . Introduction

"Network congestion is a condition where a system has settled under load into a state where traffic demand is high but little useful throughput is available, with high levels of packet loss, delay, and delay variation." – Wikipedia Encyclopedia

1.0 Overview

The Internet community started to experience a series of congestion collapses after October 1986 and researchers started suggesting ways to tackle this problem. The first available and organized literature [Jacobson 88] of congestion avoidance and control algorithms was presented at the UC Berkeley research lab. This work was motivated by the alarming situation faced by the authors when they experienced a huge rate drop from 40 Kbps to 32 bps.

TCP-based approaches were preferred and were serving well until the need for network-level computation support (Active Network) started being regarded as an important area for research [Psounis 99]. The concept of active networking emerged

in 1994 and 1995 from discussions within the DARPA (Defense Advanced Research Project Agency) community on the future directions of networking systems.

[Jain 88] states that with the era of high bandwidth network, congestion control would become a historical phenomenon; on the contrary, it is still a daunting task for networking community. [Bhattacharjee 96] regarded congestion control as a problem that was unlikely to disappear in the near future. Over the past few years the Active Networking community has been suggesting ways which highlight the importance of network level computation. The Internet experiences packet losses frequently due to congestion, but most likely the reason for this is congestion control technique [Widmer 02]. Widmer et al. state that imbalance in resource allocation should be regarded as a prime cause for congestions. [Stoica 03] praised end-to-end congestion control saying it was a *'central tenet'* of Internet architecture but in their discussion they augment the earlier argument saying that router based mechanisms greatly improve the efficiency.

"Considerable research has been done on Internet dynamics since 1988, and the Internet has grown. It has become clear that the TCP congestion control / avoidance mechanisms, while necessary and powerful, are not sufficient to provide good service in all circumstances. In addition to the development of new congestion control mechanisms, ***router-based mechanisms*** *complement the endpoint congestion avoidance mechanisms."* [RFC 2941] pp. 3.

1.1 Network Performance

If the network load is small then throughput keeps up with the load, but as it reaches the network capacity the queue starts building up and results in congestion (packets being dropped).

- ***Knee*** is the point after which the increase in the throughput is small but increase in response time is significant.
- ***Cliff*** (congestion collapse) is the point at which throughput approaches zero. At this point response time approaches infinity.

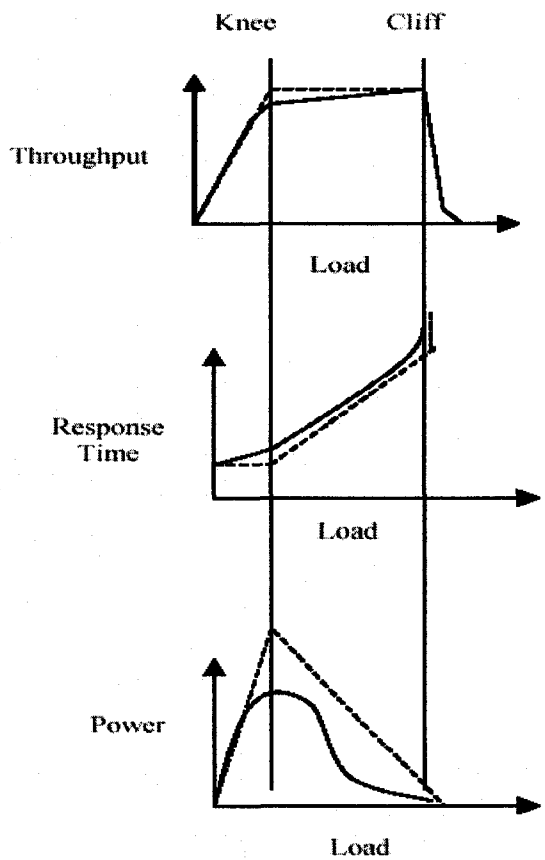


Figure 1-1: Network Performance (*Ref - [Jain 88] pp.4)*

Congestion-avoidance schemes allow the network to perform at the knee whereas congestion control schemes allow the network to operate to the left of the cliff.

Power is the ratio of throughput to response time. It can be observed that the peak of the power curve occurs at the knee response time will increase drastically from this point onwards which suggests that the main research work for improvement of congestion control techniques should revolve around quick response time and throughput.

The response time issue can be dealt with by placing congestion control capabilities into the network. Throughput issues can be dealt with as a customized flow rate and packet size combination, which will reduce the packet loss rate.

1.2 Motivation

Current congestion control mechanisms used in TCP variants adjust the packet rate in order to adapt to network conditions and obtain a throughput not exceeding that of a TCP connection operating under normal conditions in the same environment. In an environment where the bottleneck resource is bandwidth, resource sharing depends on packet size.

Current congestion control mechanisms aim to either reduce or increase the number of packets sent per time interval so as to adjust to the current level of congestion in

the network. Reducing the packet rate in times of congestion is the correct behavior in a packet rate-limited environment. In an environment where the bottleneck is limited bandwidth and different flows use packets of different sizes, the bandwidth- resources may not be shared fairly among flows. Under such a situation, the share of bandwidth of a flow may be, governed by the packet size. A TCP flow using a smaller packet size may achieve a small fraction of throughput as compared to a TCP flow with a larger packet size, if no corrective measure were taken. This fraction is, in first approximation, of the order of the ratio of the small packet size to the large packet size.

To avoid congestion on a network path, it would be sufficient for TCP to adapt to the available bandwidth at the link with the heaviest congestion and only react to packet losses that occurs at that specific router (local fairness). Since TCP reacts to all loss events regardless of where they occur, TCP behaves more conservatively than necessary when crossing multiple congested gateways [Widmer 00].

The goal of end-to-end congestion control is to adapt the resource usage of a flow to the network's available resources, and to do so such that the resources are shared fairly among competing flows. Usually, the limited resource is either the packet processing rate of the router or link capacity (bandwidth). In the former case, it is necessary to control the rate at which the packets are sent; while in the latter either the packet rate or the packet size can be adjusted. A bandwidth-limited environment therefore allows to trade off packet size for packet rate (instead of reducing the packet

rate to react to congestion, a flow may choose to maintain a high rate at the expense of a reduced packet size).

1.3 Rationale

The following figure is a schematic representation of a sender and a receiver on a high bandwidth network connected by a bottleneck link.

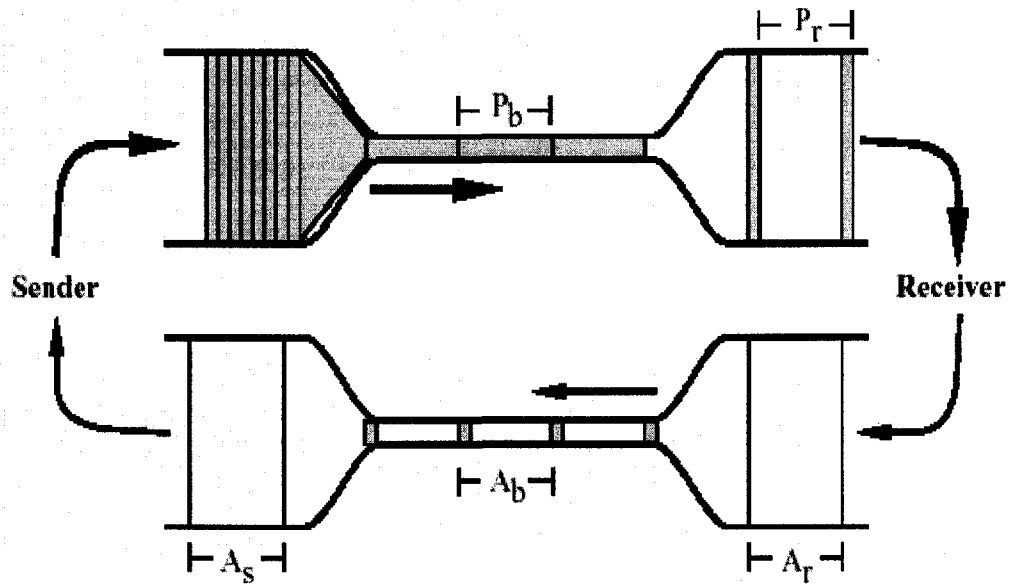


Figure 1-2: Schematic Bottleneck (*Ref - [Jacobson 88] pp.4)*

The vertical dimension is bandwidth and the horizontal dimension is time. Each of the shaded boxes represents packet. The area of the packet is the packet size.

Bandwidth * Time = Size in Bits

The time P_b represents the minimum packet spacing on the bottleneck. The net packet spacing at the receiver is P_r. P_r is equal to P_b, i.e., $P_r = P_b$. The same time delay persists in the acknowledgements received by the sender, i.e., $A_b = P_r = P_b$

The scaled packet size method presented in this work reduces the delays experienced (P_b, A_b) while the packet is in transit. Moreover the packet rate at the sender is also scaled with the link metrics (available bandwidth) measured at an intermediate node. As a net effect TCP does not have to enter the phases of slow start and congestion avoidance.

1.4 Thesis Statement

The proposed Active Congestion Control (ACC) technique presents a new intermediate router based congestion control mechanism. It makes use of available bandwidth-based congestion detection (ABCD) to judge the degree of congestion and formulates its remedy depending on it. The ACC-ABCD approach uses packet size adjustment at the intermediate router in addition to the rate control at sender.

The congestion control mechanism used by TCP reacts to a single packet loss by halving its congestion window. This causes abrupt changes in the sending rate. Our scaled rate change approach increases reliability, which is demonstrated with fewer packet drops. The ACC-ABCD approach uses packet size adjustment at the intermediate router in addition to the rate control at sender. We evaluate and validate our approach through simulation using NS-2. Our findings can be used to design a

TCP-friendly congestion control mechanism for applications that can be forced to use a small packet size.

1.5 Contributions

[Faber 98] and [Faber 02] work extended NS-2 version 2.1b8a to support the Active Network paradigm. One important contribution of the work presented in this thesis is to make the latest version of NS-2, i.e., 2.26 ACC module compliant.

The work presented here is based on a comprehensive investigation made with all TCP variants. Such studies on the dumb-bell topology have not been available. To the best of our knowledge no work has presented comparative results based on all the TCP variants.

[Bhattacharjee 96] for the first time, outlined the use of Active Networks in Congestion Control. Our approach adapts this idea with the use of Available Bandwidth-based Congestion Detection, which is helpful in measuring the degree of congestion. [Widmer 02] suggested the idea of packet size change, which is the basis of our work. TCP has a greater bias against packets that are less than MTU size, but with ACC-ABCD we claim that small size packets will be treated fairly. We infer that small size packets are sometimes a necessity of the networks especially when there is a scarcity of resources (Bandwidth) and dealing with the packet size is the right approach when the bottleneck is caused by the limitation of available bandwidth.

1.6 Thesis Organization

Chapter 1 has provided an overview of congestion control problems and the objective of this thesis. The remaining document is organized as follows. Chapter 2 provides the background of classical congestion control techniques. In the Chapter 3 we have surveyed the work related to Congestion Control involving network level computation, i.e., Active Congestion Control. Chapter 4 outlines the proposed ACC-ABCD methodology, which is analyzed in Chapter 5 with experimental details and results. The concluding Chapter 6 lists achievements and limitations of our methodology as well as future work. References used for this work have been tabulated in the bibliography section. The appendices A, B, C and D provide the quantitative and performance metrics used for network benchmarking, algorithms used in TCP variants and ABCD experiments respectively.

Chapter 2 . Background

"A network is said to be congested if it is being offered more traffic than its rated capacity. As the per-flow product of bandwidth and latency increases, TCP becomes inefficient and prone to instability, regardless of the queuing scheme." – ([Katabi 02] pp. 1)

2.0 Network congestion

"Congestion can occur when data arrives on a big pipe (a fast LAN) and gets sent out through a smaller pipe (a slower WAN). Congestion can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs." ([RFC 2001], pp. 3).

According to ([Jain 88], pp. 1) *"Congestion occurs in a network when resource demands exceed the capacity."*

There is little queuing in the backbone networks, let alone the congestion-related packet drops. To the contrary, in access networks one of the main reasons of congestion in the networks is mismatch in the speed links [Widmer 02].

2.1 Congestion Control & Avoidance

[Jain 88] distinguished between two commonly interchangeable terminologies called congestion control and congestion avoidance. According to Jain et al., congestion control is a recovery process from a congested state whereas congestion avoidance is a prevention mechanism designed to keep the network from entering the congested state. Further, they state, *"[t]he congestion avoidance scheme allows a network to operate in the region of low delay and high throughput."* These avoidance mechanisms improve the packet loss rate since they prevent networks from entering into congested state. The proposed method in this thesis falls under the category of Congestion Avoidance.

2.2 Flow Control and Congestion control

Some earlier works regard congestion control as a special case of flow control. The difference between flow control and congestion control was defined well in [Johari 01] which states that flow control is an agreement between the sender and the receiver regarding the speed of the flow, whereas congestion control speaks about the intermediate nodes which will be congested with the mismatch in various parameters. Further they state that some researchers still consider flow control to be a special case of congestion control. Johari et al. summarized it by stating, *"flow control is a bipartite agreement and congestion control is a social law."*

2.3 Active Networks

The first available literature on the notion of Active Networks, [Tennenhouse 96] characterized this paradigm as *"networks, which allow their users to inject customized programs into the nodes of the network."* The extended and comprehensive definition provided in [Tennenhouse 97] states, *"In active networks switches of the network perform customized computations on the messages flowing through them."* They further tried to investigate the need for the evolution of these kinds of networks and conclude that it's a consequence of *"user pull and technology push."* These Active Networks are a matured form of programmable networks (PNs). According to [Campbell 99] the ease of deployment of new services was the motivation for this transformation of PN to the Active Networks.

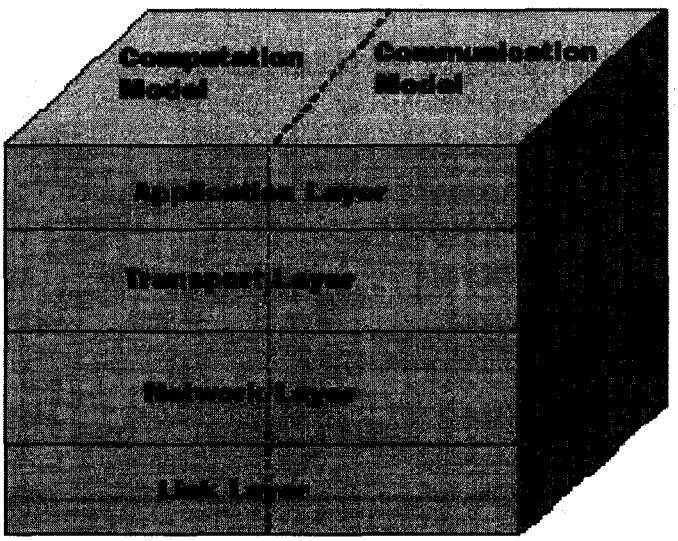


Figure 2-1: Model of programmable network *(Ref - [Huang 02] pp. 1)*

Under the concept of programmable networks, two schools of thought have emerged on how this programming support will be provided. One school is called the Opensig community, which advocated the use of 'open programmable networking interface', and the other DARPA community suggested moving away from the OSI model towards the active-network model [Campbell 99]:

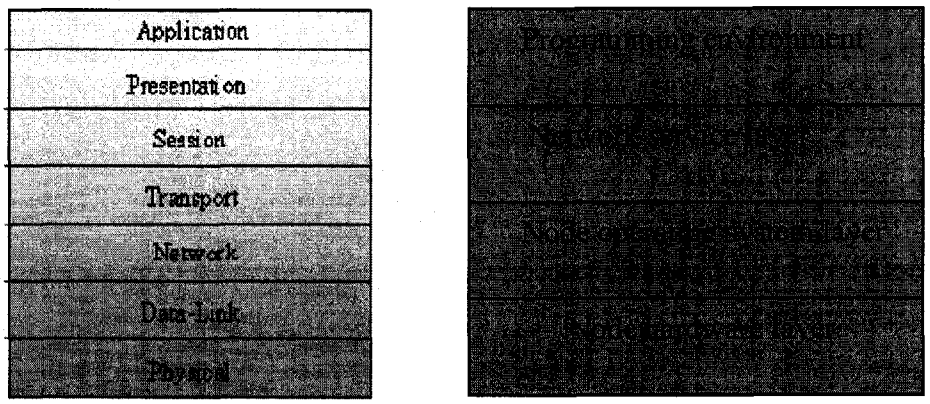


Figure 2-2: OSI model - *(Ref -[Huang 02]* pp. 2) **Active network model**

Once the networking community accepted the active-network model, it has been proposed for a wide range of applications. [Bhattacharjee 98] states the advantages of active networks as follows –

- Active networks reduce the time required to develop and deploy new network services.
- Active networks may enable users or third parties to create and tailor services to their particular applications and even current network conditions.

- A dynamically programmable-network offers a platform for experimenting with new network services and features on a realistic scale without disrupting regular network services.

2.4 Congestion Control – Protocols Perspective

As discussed in [Jacobson 88], congestion control was supposed to be taken care of by the TCP/IP suite. There are some variants of TCP such as TCP Vegas [Brakmo 95], which were needed because of certain shortcomings of TCP. Brakmo et al. discuss an end-to-end congestion scheme, which they claim to be superior to the TCP - Reno, another variant of the TCP suite. Continuing with the efforts to design better congestion-control schemes, an Active Networks paradigm was proposed [Bhattacharjee 96]. It equips the network with a capability to decide in case of congestion which data is to be dropped. [Anker 03] justifies the use of TCP saying the TCP friendliness allows applications to coexist with other TCP sessions. [Tao 03] echoes [Jackobson 88] and states that "TCP has played a critical role in the robustness of today's Internet" but also agrees with the requirement to make it better with regard to demands of QoS.

2.5 TCP – Variants

Since the first TCP implementations (1986), TCP has been improved in several ways. Earlier versions of TCP used a go-back-n retransmission scheme with cumulative acknowledgements and a retransmission timer for the recovery of lost packets. The timeout value is a multiple of the RTT. Today, many different versions of TCP are in use. This section gives a brief overview of some of the more important versions of TCP. Appendix B provides some algorithms, which will be helpful in understanding the following discussions.

- **TCP Tahoe (1988): [RFC 793]**

TCP Tahoe is one of the commonly used TCP-flavors, which is basically a TCP with Slow Start, Congestion Avoidance and Fast Retransmit algorithms. Tahoe follows a go-back-n model, uses cumulative positive acknowledgement and requires timer expiration (timeout) to resend data lost during transport. In Slow Start, the congestion window increases exponentially for each ACK received, until a threshold called Slow Start Threshold (SSThresh). Beyond that, TCP goes into the Congestion Avoidance Phase, where the window increases linearly for every RTT. In Fast Retransmit, in case of a packet loss, TCP retransmits the lost packet after receiving three duplicate ACKs (dup-acks) without waiting for a timeout. Compared to the more recent versions of TCP (Reno and New Reno) TCP Tahoe, does not have the fast recovery mechanism.

- **TCP Reno (1990): [RFC 1185]**

TCP Reno modifies Tahoe's Fast Retransmit mechanism to include Fast Recovery. This prevents the communication path from going empty after Fast Retransmit, thereby avoiding the need to Slow Start to re-fill it after a single packet loss. The sender enters Fast Recovery after receiving a threshold number of dup-acks. The sender retransmits one packet and reduces its congestion window by half. Instead of slow starting, the Reno sender uses additional incoming dup-acks to clock subsequent outgoing packets. TCP Reno improves performance when single packet losses occur, but suffers when multiple packets are lost.

- **TCP Vegas (1994): [Brakmo 95] and [RFC 2581]**

TCP Vegas is quite different from other versions of TCP in the following three ways. The first technique is with respect to rate control. Besides correctness and in-order delivery, the most important function of TCP is to avoid congesting the network while trying to reach its own maximum possible throughput. Therefore, controlling the congestion window plays an important role in determining the throughput. In other versions of TCP, e.g. Tahoe, Reno, etc., the congestion window decreases only when a loss occurs. The loss might have occurred because the network is congested. TCP Vegas, on the other hand, tries to anticipate congestion and adjusts its transmission rate accordingly.

First, TCP Vegas maintains two variables, Expected Rate and Actual Sending Rate.

Expected Rate = Congestion Window Size / Base RTT

Where Base RTT is the RTT of a segment when there is no congestion. In practice, Base RTT is set to the minimum of all measured RTT.

The Actual Sending Rate is calculated by counting the number of bytes between the time the segment is sent and the acknowledgement is received and computing the RTT for this segment.

Actual Sending Rate = Number of bytes sent in Sample RTT / Sample RTT

Second, Vegas compares Actual Sending Rate to the Expected Rate and calculates the difference to be

Diff = Actual Sending Rate - Expected Rate

It also defines two thresholds α and β. When Diff < α, Vegas increases the congestion window linearly during the next RTT. When Diff > β, Vegas decreases the congestion window linearly during the next RTT. Otherwise, Vegas leaves the congestion window unchanged.

In the second technique, Vegas uses a more accurate RTT estimate and a more timely decision to retransmit a dropped segment.

In the third technique, Vegas modifies Slow Start by allowing exponential growth only every other RTT. In between, the congestion window stays fixed so that Diff can

be calculated. This is used to switch from Slow Start to linear increase/decrease mode.

- **TCP SACK (1996): [RFC 2018]**

TCP SACK uses a selective repeat scheme to retransmit lost packets. The receiver sends back selective ACKs to the sender on receiving out of order segments. The sender then selectively retransmits the missing segments. The other versions of TCP listed above, require modifications only to the TCP sender. But SACK must be implemented at both the sender and the receiver.

- **TCP NewReno (1999): [RFC 3782]**

TCP NewReno modifies the Fast Recovery algorithm in Reno to respond to the partial acks received during Fast recovery. A partial ack is an ACK for the retransmitted packet that acknowledges only some but not all the packets that were transmitted before the Fast Retransmit. TCP Reno goes out of Fast Recovery and waits for a timeout if partial ACKs are received during the Fast Recovery phase. In TCP NewReno, partial acks do not take TCP out of Fast Recovery. Instead, partial acks received during Fast Recovery are treated as an indication that the packet, immediately following the acked packet, has been lost and should be retransmitted. Hence, TCP New Reno performs better with multiple losses.

- **Delayed ACK Option: [RFC 2883]**

Delayed ACK is an option that can be used at the TCP receivers. Delayed ACK allows a receiver to refrain from transmitting an ACK for every incoming segment. The receiver could send back an ACK for every alternate segment. Since ACKs are cumulative, delayed ACK does not reduce the transmission reliability. Also, it conserves resources by decreasing the load on the network and the machines that must generate and process these segments.

2.5.1 Mechanisms which Complement TCP

- **TCP through RED gateway:**

Random Early Detect (RED) is implemented in gateways or routers in the network. The gateway detects onset of congestion by measuring the average queue size, and then occasionally drops a packet to notify TCP sources of the possibility of congestion.

- **TCP with ECN Option: [RFC 2481]**

The Explicit Congestion Notification is a congestion avoidance mechanism like RED, and is normally used with RED to detect the onset of incipient congestion, and force TCP to reduce its congestion window before a loss actually occurs. The RED

gateway, on detecting congestion, sets the ECN bit in the IP header of the packet, and instead of dropping it, forwards it to its destination. The TCP receiver, on receiving the packet, sets the ECN bit in the ACK packet back to the sender. The sender on receiving this packet with the ECN bit set, reduces its congestion window, thus reducing its sending rate.

2.6 Survey of Congestion Control Techniques

2.6.1 Premature Stage

Congestion control and avoidance was well documented for the first time in [Jacobson 88], which incorporated into the TCP suite some of the congestion-control remedies like slow start amd congestion avoidance. More details on these algorithms are available in appendix B. [Jain 88] emphasized the difficulties in congestion control in the connectionless network layer. As referred to earlier, one of the contribution of the [Jain 88] work was to differentiate between flow control and congestion control, which has proved to be a milestone contribution in this field.

2.6.2 Later Developments

After early exposure in the late eighties the network community prioritized the congestion-control problem. [Lotfi 93] explored one more reason of congestion control which they summarized as *"[c]ongestion is a mismatch of available resources and the amount of traffic."*

[Brakmo 95] proposed an end-to-end congestion avoidance scheme (which later came to be known by the name TCP Vegas), that increases the throughput and decreases the packet loss. To be exact, they assert that the experiment on the actual Internet and in simulation achieves 47 to 71 percent better throughput and one fifth to one half of the packet loss. Then [Golestani 98] formulated the problem of end-to-end traffic as an optimization problem. With this, the authors claimed to have developed a methodology that satisfies the congestion avoidance norms as well it satisfies the users regarding fairness in the process.

[Yang 95] summarized the work done in these two time spans by presenting a taxonomy of congestion control techniques based on control theory. Later it was used for the development of new congestion- control strategies. The basic characteristic used for the classification is how an algorithm collects & interprets the information from the feedback. In short, it focuses on the decision-making process.

2.7 Classical Techniques

2.7.1 ECN marking

Explicit Congestion Notification (ECN) is used as a means of conveying congestion information back to the end systems. ECN was proposed in 1999 by [RFC 2481] and incorporated into the TCP/IP suite in 2001 by [RFC 3168]. The researchers have been discussing the enhancement of ECN, which will provide the QoS support. Also it is considered as a need of today's networks." [Kunniyar 03] proposed a scheme to adjust the ECN marking such that, a loss of packets will be minimal. [Li 03] talks about TCP-ECN where bottlenecks mark the packet so that packet loss is avoided.

2.7.2 Fair Queuing

With regards to end-to-end argument, Fair Queuing controls the congestion in gateways by restricting every host to an equal share of gateway bandwidth. [Legout 02] proposed a multicast congestion control protocol (called PLM) for audio/video and file transfer applications based on a cumulative layered multicast transmission, which is compatible with the TCP suite. In addition to this the authors also demonstrate how to apply the FQ paradigm for designing new congestion-control protocols. They claim to have separated the requirements of the network from the requirements of end system.

2.7.3 Rate-Based Schemes

TCP is equipped with the AIMD (additive increase multiplicative decrease) mechanism, which is mainly responsible for rate control. [Rejaie 99] presented a comparison between the Rate Adaptation Protocol that employs AIMD and TCP. [Li 03] said that in traditional congestion-control schemes, the packet loss is unavoidable. Li et al. suggests a new strategy where they refer to their earlier work [Harrison 01] for a threshold technique and the new concept of accumulation. This approach doesn't require packet marking. It tends towards the rate adaptation technique. [Subramanya 03] presented a rate based congestion control scheme. This scheme is modeled on non-linear system theory by which performance and stability can be mathematically analyzed and established. Their results confirm the mathematical claim with a simulation result.

2.7.4 Window-Based Schemes

TCP [RFC 793] uses a window-based scheme for congestion management. [Jacobson 88] describes the dynamic window sizing on congestion. [Mo 98] presented the multiclass closed-fluid model, which is used to justify the existence of a window-based fairness scheme for end-to-end congestion control algorithms. The theory presented in this work is supported by mathematical proof but a practical implementation is yet to be achieved.

2.7.5 Random Early Detection (RED)

[Floyd 97] states that RED gateways improve the fairness and performance of TCP traffic. They have presented a RED queue-management scheme, which keeps the buffer from overflowing. They infer that RED configuration is still a research issue. [Janarthanan 03] states that RED gateways are intended for a network where a transport protocol responds to congestion indications from the network.

2.8 Feedback-Based End-to-End Congestion Control

2.8.1 What are they?

Feedback based congestion controls are common in network protocols. Widely deployed byte-stream protocols include the Internet Transmission Control Protocol [RFC 793]. [Faber 98] states that network systems have to change themselves dynamically to adjust the responses according to the status of the congestion site. In this regard [Rejaie 99] states *"[a]ll end systems are expected to react to congestion by adapting their transmission rates, to avoid congestion collapse and to keep network utilization high."* [Mo 98] states the advantages of these systems saying only the hosts that want to implement different functions need to upgrade their software and this way the network stays simple and it scales easily.

2.8.2 Some of the Approaches

Congestion control has been traditionally dependent on the end-to-end adjustment of the flows. Sources adjust their transmission rate in response to feedback information from the network nodes [Lotfi 93].

[Golestani 98] proposed a minimum cost-flow control algorithm. The authors implement a number of new schemes in the TCP such as a timeout mechanism, control of number of extra buffers, and a modified slow start mechanism. They also claim that it does not affect the latency and it is comparable with TCP-Reno in all aspects. [Psounis 99] describes the scheme where centrally managed techniques will be replaced by distributed network management. Network management is achieved through the polling of managed devices, which are observed for anomalies. Managed devices are responsible for feedback. [Floyd 99] describes the idea of a router taking the initiative and restricting the much demanding flows at the time of congestion. This idea gives rise to use of network level computation. The authors also raise questions about the credibility of the best effort service that is non-congestion controlled. [Santos 03] proposed a packet-marking scheme in which end-to-end congestion control is achieved, which is link-level controlled and it prevents congestion from spreading. It defines the response functions based on feedback in case congestion is detected.

2.9 Deficiencies

[Faber 98] utters a word of caution saying that feedback-based congestion-control systems do not scale well enough with respect to network bandwidth and delay. Further the authors explain that, increase in those quantities decouples the congestion site and the end point. [Floyd 99] states that duration of congestion at the bottleneck of connection is directly proportional to the bandwidth delay product.

2.10 Summary

In this chapter we reviewed different versions of TCP. The chapter presents a taxonomy of congestion control algorithms based on their functionality. In the last section we looked at the feedback based congestion control techniques to define the need for improvement of these techniques was being considered by researchers. In the next chapter we shall present the state of the art: Active Congestion Control.

Chapter 3 . Related Work

"Active networks allow individual user, or groups of users, to inject customized programs into the nodes of the network. 'Active' architectures enable a massive increase in the complexity and customization of the computation that is performed within the network, e.g., that is interposed between the communicating end points."
[Research Group at MIT]

3.0 Internet to ActiveNet

3.0.1 Early Research (What are Active Networks?)

[Tennenhouse 96] proposed the preliminary idea of Active Networks in the form of RFC named 'From Internet to Activenet', which basically speaks about the incorporation of Activenets, which will take over today's Internet. [Tennenhouse 97] envisioned this field in the mid 90's when the need for network-level computation was felt. This was initially part of discussions between DARPA communities. In this

very first published work of Active Networks, Tennenhouse et al. outlined two different approaches to Active Networks –

- Discrete Approach – Users would first inject their custom processing routines into the required routers. Then they would send their packets through such "programmable" nodes. When a packet arrives at a node, its header is examined and the appropriate action is taken according to program.
- Integrated Approach – This is a more extreme view of active networks in which every message is a program. Every message (called *'Capsule')* that passes through nodes contains a program fragment that may include embedded data. When a capsule arrives at an active node, its contents are evaluated.

Active Network was mainly a brainchild of DARPA (Defense Advanced Research Project Agency). The ABone is a virtual testbed for the active networks research program funded by DARPA. It is composed of a set of computer systems configured into virtual active networks. These systems execute under UNIX -- FreeBSD, Linux, or Solaris -- or operating systems built specifically for active networking.

3.2 Network level Computation

The very first work to speak about the network level computation issue is [Bhattacharjee 96], which extends the active-networks application scope to congestion control. They emphasize the application-specific processing of data. [Floyd 93] had in an earlier work commented regarding possible use of programmable networks in network management. [Psounis 99] also discussed the idea of network-level management, which the authors believe will reduce the traffic of redundant information about network status. [Schwartz 00] made one of the first attempts to apply the Active Network paradigm to network management and monitoring. The approach is to move management stations close to the nodes to be managed. The advantage of using custom packets, which are executed at these nodes, is that management stations can get useful information rather than redundant and useless information. These smart packets follow the integrated (capsule based) approach towards using the active-network paradigm

3.3 Rationale for the Use of Active Networks in Network Management

In conjunction with the proposal to use active-network technologies for congestion control, [Bhattacharjee 97] proposed that the end-to-end system may be used as a scale to judge the network-level computation.

3.3.1 End-to-End Argument

The end-to-end argument provides a rationale for moving the function in a layered system closer to the application that uses the function. The definition provided by [Bhattacharjee 97] is as follows: *"[f]unctions should be placed in the network only if they can be cost-effectively implemented there."* This work further strengthens the bonds between the end-to-end argument and the Active Network by stating *"[a]ctive networking is a natural consequence of the end-to-end argument, because certain functions can be most-effectively implemented with information that is available inside the network."*

3.2.2 Congestion Control as a function of Network Management

Traditionally, network management has been a centrally controlled phenomenon. Management stations routinely poll the devices that are under their management. [Psounis 99] demonstrated how centrally located management stations initiate a large amount of traffic (congestion), which can be suppressed, with the suggested technique of active networks. Thus, congestion control was regarded as a necessary part of efficient network management. [Stadler 02] and [Kawamura 00] present a distributed network-management scheme, which in turn takes into consideration the issue of congestion control. Thus the network community started regarding congestion control as an integral part of network management. [Raz 00] clearly stated that congestion control is a special case of network management.

3.4 Active Networks for Congestion Control

[Bhattacharjee 96] predicted for the first time that the active network paradigm would be suitable for solving the problem of congestion control. Some of their assertions -

- Active networking in its most general form will require substantial changes in network architecture. To move the network in this direction there must be some benefits, even with partial implementation.
- Transmission bandwidth and computational power will both continue to increase, but so will application requirements for bandwidth. In particular we expect that network node congestion will be due to bandwidth limitation.
- Applications would prefer to adapt their behaviour dynamically. There will be times when networks will reject the requests for bandwidth and reserved bandwidth is likely to cost more.
- Another well-known challenge of sender adaptation is detecting an increase in available bandwidth.

Only a limited set of functions can be computed at an active network node. Computing these functions may involve state information that persists at the node. Congestion is an intra-network event and is potentially far removed from the application. The time that is required for congestion notification information to propagate back to the sender limits the speeds with which an application can self regulate to reduce congestion or ramp up when congestion has cleared. They proposed a range of schemes for processing application specific data such as unit level dropping, buffering, and rate control.

- Unit level drop – A packet that specifies one particular function (one particular flow) is dropped; for some time thereafter, every packet that matches the same subset of its labels is also dropped.
- Buffering and rate control – Putting additional buffering into the switch, which has to monitor the available bandwidth and rate control the data.
- Media Transformation – It's an intelligent dropping of data when congestion occurs. Transformation of data at the congestion point into a form that reduces the bandwidth but preserves as much useful information as possible.

[Faber 98] and [Faber 02] – Active Congestion Control (ACC) uses router participation in both congestion detection and recovery. Congestion is detected at the router, which also immediately begins reacting to congestion by changing the traffic that has already entered the network.

Incorporating congestion detection as well as recovery at the router reduces the feedback delay. In feedback based end to end systems, congestion recovery starts at the end points and in turn propagates into the network whereas in this approach it starts at the congestion point and propagates to the end point. This lends a much greater stability to the system.

Feedback based congestion control systems do not scale well with respect to network bandwidth and delay because increase of either quantity decouples the congestion site and endpoint.

The ACC systems works as shown in figure 3-1:

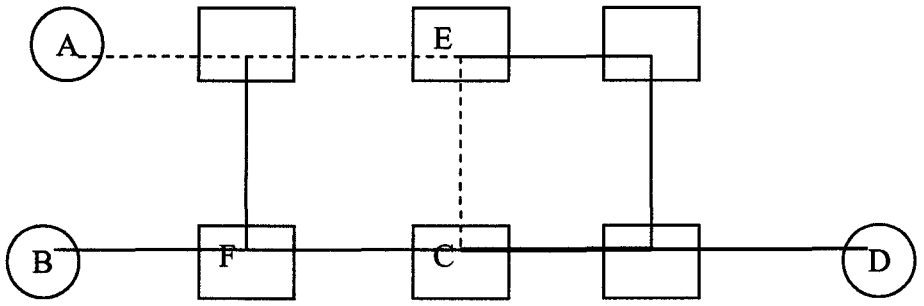


Figure 3-1: Faber's ACC model *(Ref - [Faber 98] pp. 2)*

Router C has been programmed by the first packet of the connection with instructions on how to react to congestion and subsequent packets include information on the current state of the endpoints congestion control algorithm. The router then installs filters at C itself or at neighbouring routers E or F depending on which end point is responsible for congestion. It will filter out the traffic, which it would not have sent if the congestion was detected. ACC uses Active Networking technology to greatly reduce the control delay that the feedback congestion control system exhibits.

- The current state of the endpoint's feedback algorithm is included in every packet. This state is small, an integer or two to keep the packet overhead low.

- When a router experiences congestion and it is forced to discard the packet, the router calculates the new window size that the endpoint would choose if it had instantly detected the congestion.
- The router then deletes the packets that the endpoint would not have sent and informs the endpoint of its new state.

ACC moves the endpoint congestion control algorithms into the network where they can immediately react to congestion. ACC routers instantly unsend packets that would have prolonged congestion in the network. Internal network nodes beyond the congested router see the modified traffic, which has been tailored as though the endpoint has instantly reacted.

Figure 3-2 shows the actual simulation topology used for ACC experiments. (Same topology has been used for ACC-ABCD evaluation.)

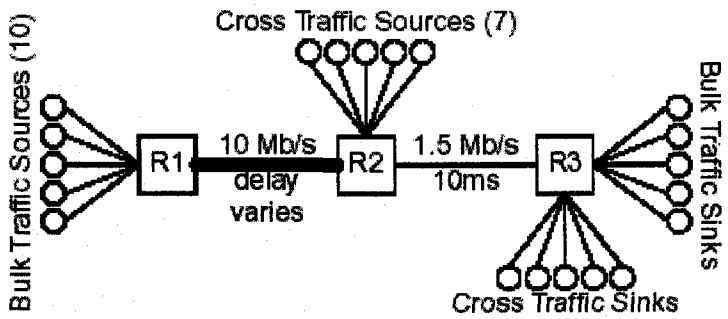


Figure 3-2: ACC topology *(Ref – [Faber 02] pp. 4)*

ACC reacts more quickly to congestion than a conventional feedback system can, which reduces the duration of each congestion episode. In this way fewer endpoints experience congestion during each episode, which improves the connection throughput.

[Lemar 99] adopted a proven TCP congestion control scheme in Active Networks, while leaving out unnecessary facets and adopting interesting ideas. Lemar et al. designed and implemented a reusable congestion control component used as a part of protocol capsule type definition. When a capsule is created and sent, the sender node evaluates the capsules forwarding method, which requests to use the congestion control method while forwarding. At the receiving node the forwarding method is evaluated for the last time and the capsule requests to send the ack for the last time. Lemar et al. have also addressed the issue of losses due to true loss instead of congestion.

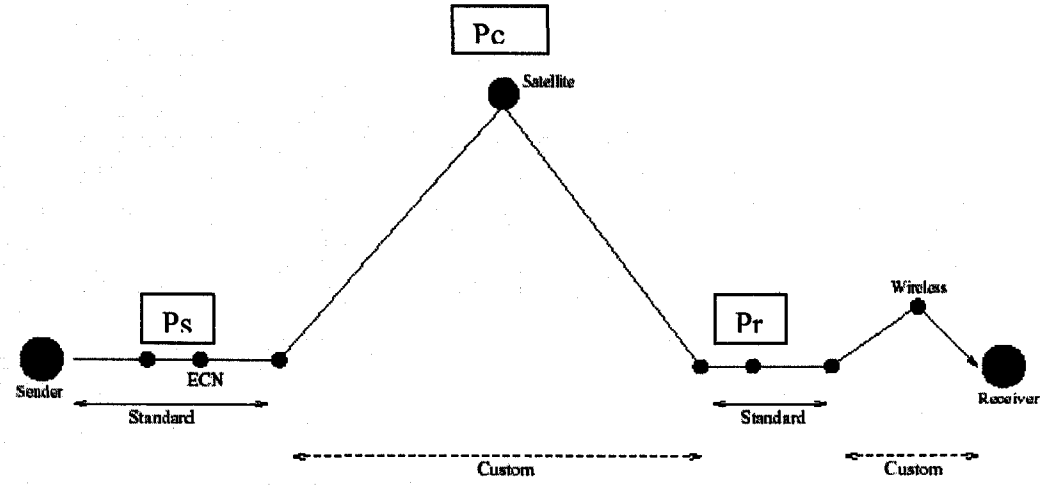


Figure 3-3: Path with many unusual links *(Ref – [Lemar 99] pp.12)*

Sender S and receiver R are on the opposite sides of a wireless link. There are two active routers Ps and Pr on the receiver and sender side, respectively. If both of them take up the job of acking, then both want the capsule to ack back to it. They solve this deadlock situation by delegating this job to a router PC which is basically a router situated at the wireless link and it knows that Ps is a special router. It solves one more problem that Pc indicates packets loss by congestion only, not the one because of noise in the wireless loop.

[Wang 00] demonstrated that the FACC (forward active network congestion control) algorithm greatly reduces the congestion reaction delay of source end points. The authors presented an algorithm and performance analysis based on simulation results. The authors also proved that traditional use of Tahoe TCP at the end point for the detection of congestion is no longer efficient and strongly support the claim for moving the computations into the network with the FACC algorithm. It compares FACC (forward active network congestion control) with Tahoe TCP and demonstrates how FACC makes feedback congestion control more responsive. FACC increases the average throughput of each end point with or without cross traffic.

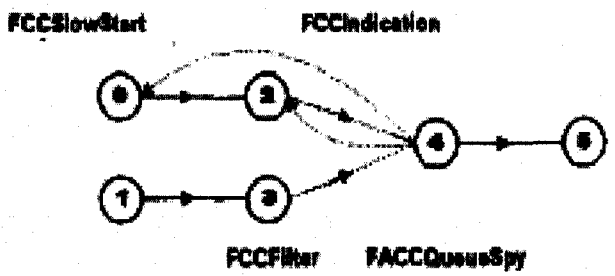


Figure 3-4: Topology of FACC *(Ref – [Wang 00] pp. 2)*

In figure 3-4 a module (FACCQueueSpy) monitors the queue of the network node. It sends FACCIndication message to upper-flow and source end point requesting them to control the data flow respectively. FACCFilter is responsible for responding to FACCIndication. FACCSlowStart employed in source end points is utilized to react to FACCIndication.

[Gyires 00] proposed an active network algorithm that can reduce the harmful consequence of congestion due to aggregated bursty traffic. When traffic burstiness at a router exceeds a certain threshold and the routers buffer size, the router divides the traffic into independent paths. When traffic burstiness falls below the threshold the dispersed paths are collapsed into the original single path. The number of paths depends on the burstiness. The paths are going to be the best adjacent routers, which are decided by historical experience (that have proven to be capable of taking over dispersed traffic in previous cases with minimal cost). Cost includes number of dropped packets etc.

[Cheng 01] presented a network assisted congestion control (NACC). NACC utilizes RTCP packet as the control message carrying information about the desired transmission rate to use. The intermediate routers adjust this value according to available resources and forward the control messages to the next node. Receiver transmits the updated information and the sender adjusts transmission behaviour according to the received information.

[Raz 01] presented a bottleneck detection technique. They claim in today's IP networks there is only one ad-hoc technique to examine one specific QoS parameter, namely delay along the path (figure 3-5).

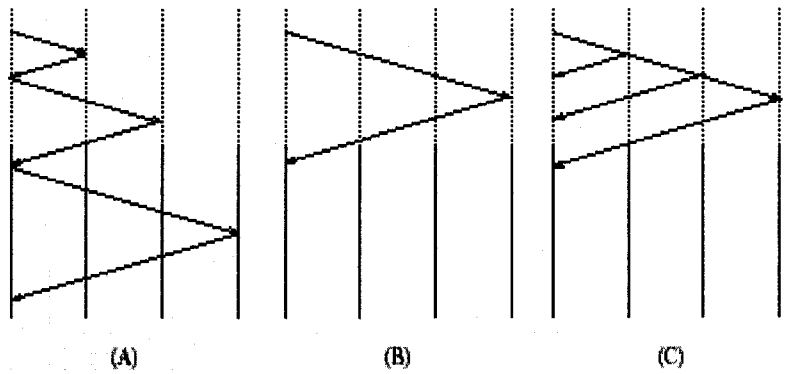


Figure 3-5: Probing techniques *(Ref – [Raz 01] pp.6]*

The current probing technique (a) sends a probe packet each time by an increased time-to-live value so that it would traverse the entire network hop by hop. They have used two different packet-probing techniques namely –

(b) Collect-en-route – Send a single packet that will traverse the route and collect the desired information from each active node. When the packet arrives at the destination, it sends the data back to the source

(c) Report en route – Send a *single* packet that will send the information back to the source when it arrives at the node and forwards itself to the next node.

Both programs can collect any desired datum from a router for bottleneck detection statistics about TCP packet loss along a route to a certain host in order to identify the bottleneck.

[Stadler 02] presented an example of application-level active networking. They used mobile agents for implementing decentralized control of management tasks. Stadler et al. proved the limitations of centralized (Manager-Agent) control management such as poor scalability and large amount of traffic. They used navigation patterns to separate the flow of control messages. This navigation pattern has embedded distributed graph traversal algorithms, which dictate its terms on the control scheme for a particular pattern.

[Xicheng 02] presents a scheme, based on the coordination of core and edge based routers, that is capable of handling responsive as well as unresponsive traffic which is called generic congestion control. When a router detects congestion it notifies the upstream router for rate control. Packet loss would be avoided if the notification is made early enough. Therefore an early detector and efficient rate controller is needed. Some lasting congestions can be reported to the host. Congestion detection is at the queue level. The RED algorithm is used for congestion detection and feedback. It can detect incipient congestion by calculating average queue size.

3.5 Available Bandwidth Estimation

[Cheng 03] used a three packet-probing algorithm for determining available bandwidth for a layered multicast congestion control. This technique was first proposed in [Cheng 01].

Packet pair is a promising approach to estimate bandwidth in a multicast environment. Packet pair relies on the fact that if two packets are queued next to each other at the bottleneck link, one is t second apart from the other after traversing the bottleneck link. t is calculated as

$$t = PacketSize / B_{bottleneck} \quad (1)$$

where *PacketSize* is the size of the second packet, and *B*bottleneck is the bottleneck bandwidth as illustrated in Figure 3-6:

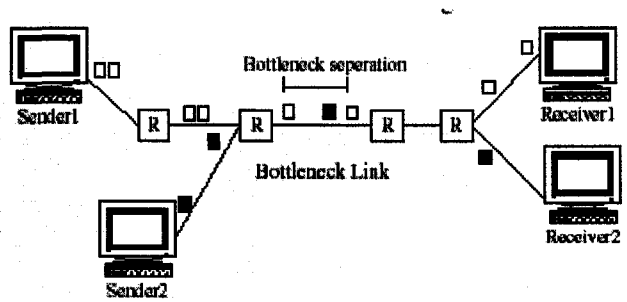


Figure 3-6: Bottleneck example *(Ref- [Cheng 01] pp.3)*

Receiver Based Packet Pair (RBPP) and Sender Based Packet Pair (SBPP) are two typical types of packet pair algorithms, which measure the bottleneck bandwidth by arrival times of packets and acks, respectively. In general, the results of SBPP may be highly inaccurate during congestion.

Multicast applications have used the RBPP algorithm to measure the bottleneck bandwidth for each receiver.

In this approach, they send three packet-pair probing packets (back-to-back, which is denoted as a burst) for the entire multicast groups every second. Taking the channel efficiency into account, particular control packets for probing are avoided; instead, only data packets are used in our work. All multicast groups of the same session are considered as a single flow. The receiver can estimate the available bandwidth by –

$$B_{available} = \frac{PacketSize_2 + PacketSize_3}{t_{a3} - t_{a1}}, \qquad (2)$$

where *PacketSize2* and *PacketSize3* are the sizes of the second packet and the third packet, respectively, and ta1, ta3 are the arrival times of the first and the third packet, respectively.

3.6 Summary

In the first section of this chapter, we explained the proposed transition from the Internet to Activenet. In the next section we stated why researchers feel the necessity of network level computation and described how the congestion control problem benefited from this new paradigm of Active Networks. In the subsequent section we have described the work done in the field of active congestion control. Our proposed method is based upon the research work described in this chapter. The method of available bandwidth calculation has also been discussed in this chapter. In the next chapter we shall explain the proposed ACC – ABCD.

Chapter 4 .
Proposed Methodology (ACC – ABCD)

"TCP throughput scales linearly with the packet size (half the packet size results in half the throughput). This behavior is reasonable if packets "cost" the same independent of their size. However, if the only scarce resource in the network is bandwidth, then it makes sense to send with the same bitrate as a TCP flow with MTU sized packets but with a higher packet rate in case the packets are smaller." – [J. Widmer on Postel Centre discussion group]

4.0 Active Congestion Control (ACC)

Placing the whole responsibility of congestion control on the end nodes raises several issues such as slower reaction to congestion, unfair allocation of network resources, and, most important, congestion collapse [Maimour 02].

If there is help from the network, faster congestion relief can be attained. Fair and accurate decisions can be made after congestion detection.

Current congestion detection is broadly based on the packet loss. We are trying to alleviate this situation by using Available Bandwidth to diagnose the congestion, that is about to happen.

4.1 Available Bandwidth-based Congestion Detection (ABCD)

The concept of available bandwidth has been of central importance throughout the history of packet networks in both research and practice. [Jain 02]

The available bandwidth is defined as the maximum rate that the path can provide to a flow, without reducing the rest of the traffic. As the utilization of bottleneck link increases, available bandwidth decreases.

In our approach to use the available bandwidth as a means of congestion detection we have chosen a non-intrusive technique, i.e., routine data packets do the job of probing the network.

We interpret the available bandwidth definition as the amount of data that can be inserted into the network path at a certain time, so that transit delay of these packets would be bounded by a given maximum permissible delay.

4.2 Rate Estimator

The ABCD approach requires that the rate estimation should incorporate the packet size, because dropping probability is independent of packet size and depends only on rate and its share of bandwidth.

Throughput is directly proportional to packet size, hence, a perfect scaling of rate and packet size should help maintain a high throughput in case of congestion.

Traffic rate estimator used in our system converged to actual rate under a wide range of network conditions.

4.3 Buffer Management

RED queues are capable of notifying senders about the growing congestion levels ahead of the time. This queue size can be measured in packets or in bytes. The congestion is detected by number of packets arriving or in terms of the size of those packets, respectively. There are trade-offs to measuring the queue in terms of bytes or packets.

In our system we make use of byte mode. The queue size is fixed to a certain number packets but the mean packet size is set to the original size of packets. So the buffering capacity of the queue to store the incipient packets is in bytes not in packets, hence it will adjusts the incoming packets as their size would be less than the original size.

4.4 Problems of current Congestion Control techniques

In our background study of current TCP based congestion control techniques we identify following shortcomings. Considering these shortcomings we model our ACC – ABCD system:

- Determining the available capacity in the first place.
- Congestion is determined only after packet drop.
- Packet drops caused by any reason other than congestion also receives the same remedy, i.e., slow start
- Packet drop at the start of slow start deteriorates the performance even more.
- Adjusting to changes in the available capacity. (Sometimes networks are under utilized)

4.5 ACC – ABCD

Step 1. Monitor the link where bandwidth is the bottleneck resource (Three packet theory).

Triplet of data packets traverses the monitored link. Available bandwidth is calculated at the receiver router (node at the other side of the monitored link) using the formula

$$B_{available} = PacketSize_2 + PacketSize_3 / t_3 - t_1$$

$PacketSize_{i\ (i=1,2,3)}$ = packet size in bytes

$T_{i\ (i=1,2,3)}$ = arrival time of packet in seconds

At any bottleneck link, when two packets are queued next to each other the separation factor 't' is enforced because of the scarcity of bandwidth 't' is given by –

t = PacketSize / Bandwidth - **[Jacobson 88]**

t is directly proportional to packet size so we choose the reduction in size.

Step 2. Here the link under consideration is facing the scarcity of link bandwidth. Hence the available bandwidth is the bottleneck resource. We scale packet size linearly by observing change in the available bandwidth. We have chosen the threshold of 0.5 Mb/s; at any value below this we start reducing the packet size by 200 Bytes for each 0.1 Mb/s.

In our dumbbell topology this particular bottleneck link was observed since we wanted to study the behavior of the system in different bandwidth delay product. Also the threshold of 0.5 Mb/s was chosen by studying and observing the trend of the system in different situation where we found that as available bandwidth is dropping below 20 percent, it is going towards congestion.

Step 3. The queue at the sender router is measured in bytes. That means the upper bound of this queue buffer is (number of packets * mean packet size). Each de-queued packet is evaluated against the packet size that (step 2) has been calculated to be appropriate.

Step 4. Rate needs to be scaled in accordance with the new packet size. We estimate this rate change using –

$$r = (1 - e^{-T/K})\, S / T + e^{-T/K}\, r_{old} \qquad \textbf{[Stoica 04]}$$

Where -

$$\mathbf{T} = \Delta t = (\delta t_2 - \delta t_1 / 2)$$

δt = enqueue (t) – dequeue (t) i.e. inter arrival time.

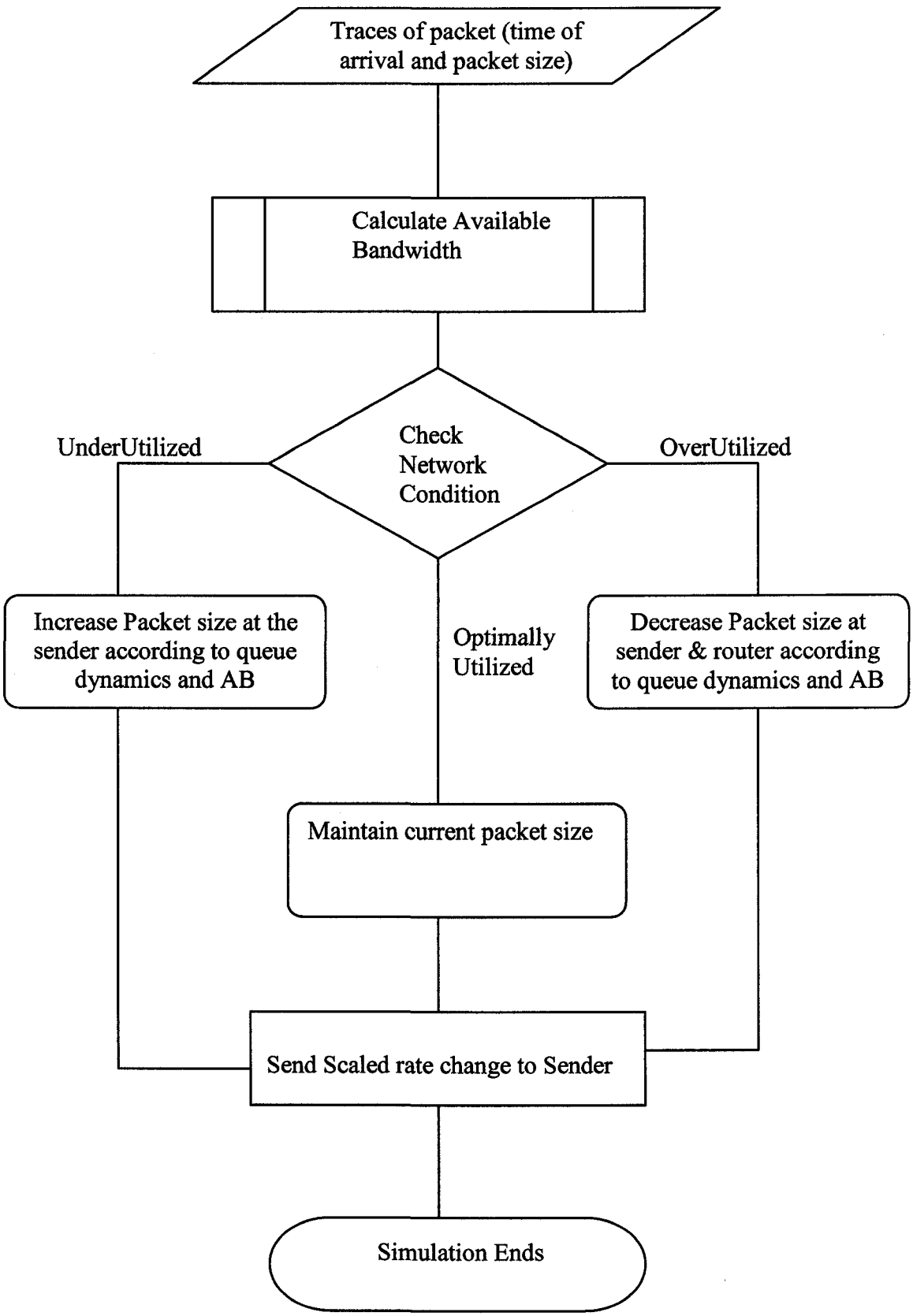
S = Change in the size of the packet, **K** is a constant (between 100 and 500)

The spacing of two packets (δ t) at the queue provides a measure of an amount of traffic in the link i.e. congestion level.

When the link is congested, the fair rate r is computed such that the rate of the aggregate incoming rate equals the link capacity.

Step 5. This process of observing drop or increase in available bandwidth continues for each triplet received at the other end of the link under consideration.

ACC – ABCD Flow of Events



4.6 ACC – ABCD Illustration

- **Scenario 1**

To understand the functioning of our ABCD system we will again consider the ACC system discussed in chapter 3 (figure 4-1)

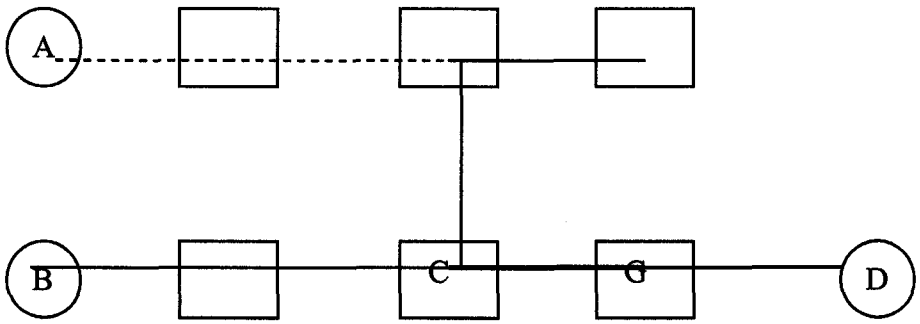


Figure 4-1: ACC – ABCD Illustration 1

Sender A and B are two flows sending packets to destination D via router C. Link C to G is the bottleneck of the network. The available bandwidth measurement module is monitoring the link C to G and sending the feedback to router C.

Router C is capable of taking an action with packet size according to the available bandwidth feedback from G. When available bandwidth falls below the threshold, the router C reduces the size of the outgoing packets and sends a feedback to the respective senders telling them to adapt to the changed environment.

If sender A's packets were reduced in size, the feedback of scaled rate and new packet size will be sent to A. By the time A receives this information whatever data it has already sent will be taken care of at router C.

In cases like this A might be sending packets of different size than B, but both the flows are treated equally. Also C takes care that whatever traffic enters the link C to G is in the best interest of the network.

- **Scenario 2**

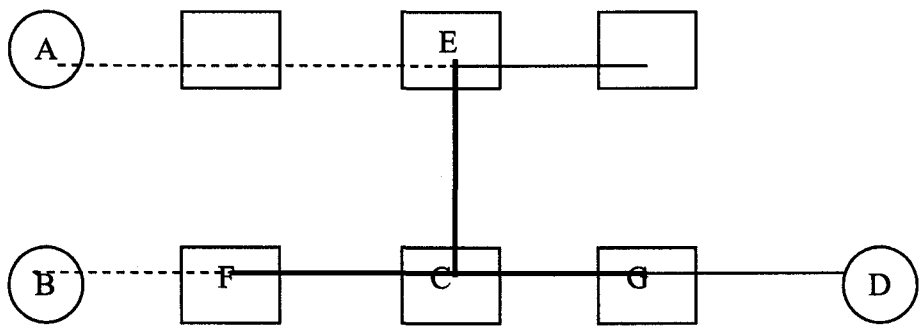


Figure 4-2: ACC – ABCD Illustration 2

In scenario 2 (figure 4-2) links EC, FC and CG all are monitored for the change in bandwidth. Routers E, F, and C all are capable of taking an action according to the condition of the respective outgoing links. In such case traffic would be altered at E and F only so that point C is least likely to face a problem of congestion. Even if

other traffic sources (not shown in the figure) make the link CG congested, the degree of congestion will not be as bad as it would have been without check point E and F.

With the above two scenarios we claim that the ACC – ABCD system will work efficiently with congestion caused by responsive cross traffic as well as non-responsive cross traffic.

The system would be considered stable because the changes are carried out from the point of congestion to the sender. It means that the role of all the routers will be significant as compared to passive networks where only the end nodes (A and B in this case) are responsible for these kinds of actions.

It is observed that incoming packets that were delivered by an end node before it-got the congestion notification make the congestion situation worse. Since it will be avoided in our system, we claim that it will outperform traditional congestion control systems.

4.7 Summary

This chapter has described the proposed ACC – ABCD system. In the next chapter we will verify our assertions with the experiments in a simulation environment. A wide range of experiments conducted by us will help in understanding the functioning of our system.

Chapter 5 . Experiments, Results and Analysis

"A simulation is the execution of a model, represented by a computer program that gives information about the system being investigated. The simulation approach of analyzing a model is opposed to the analytical approach, where the method of analyzing the system is purely theoretical. As this approach is more reliable, the simulation approach gives more flexibility and convenience." – [Simulation Model Book by J Banks]

5.0 Simulation Environment

In order to evaluate ACC - ABCD, we have implemented it in a NS-2 network simulator and measured the performance of ACC – ABCD under various conditions.

NS-2 is an open-source simulation tool. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks.

NS (version 2) is an object-oriented network simulator written in C++ and OTcl. Simulation topologies are written in TCL (tool command language) language, which are linked with background C++ simulator modules using Otcl linkage. NS is primarily useful for simulating local and wide area networks.

For analyzing the results we have used scripting languages like perl and awk. The NS-2 writes all traces to a file, which needs to be parsed to present the results in the desired manner. Analysis tools such as Trace-graph and NANS-2 have been used for graphical analysis of results. Network Animator (NAM) provides animation of the simulation.

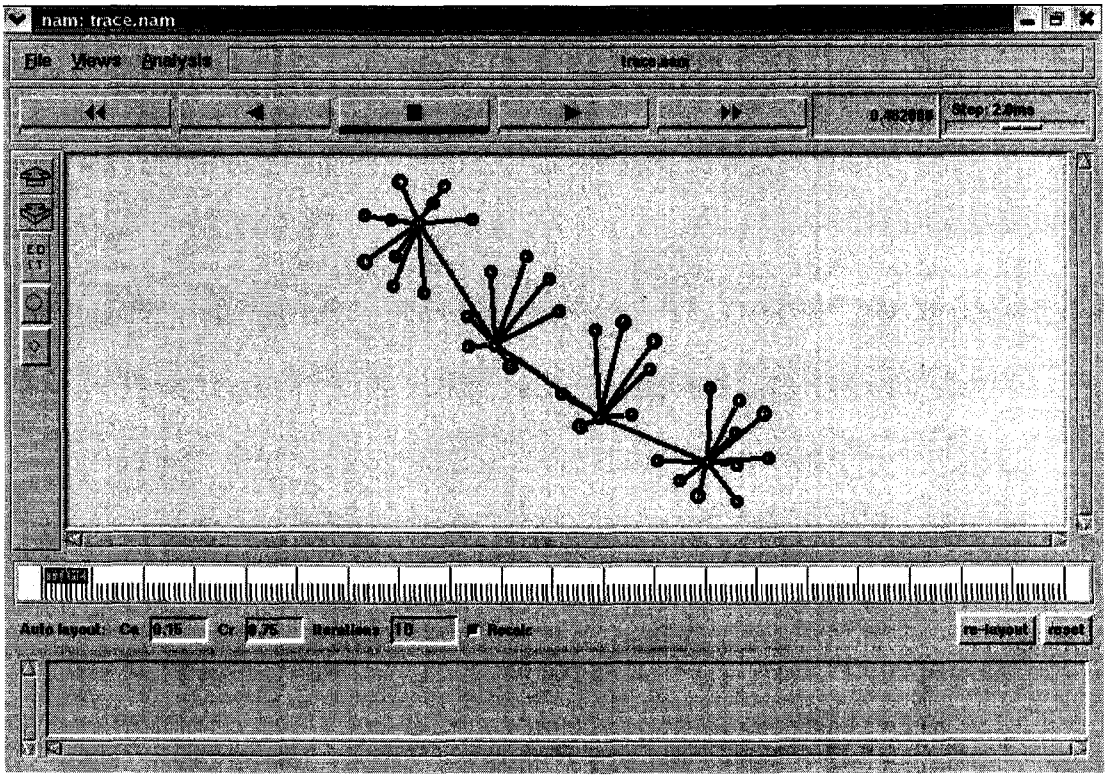


Figure 5-1: Screenshot of Simulation (NAM)

Traditionally NS-2 does not support the Active Network paradigm; we have carried out changes in the NS architecture so that we could implement the customized modules into the NS-2.

For initial topology generation we have used a tool called 'Nscript' developed by George Washington University. It provides a graphical user interface for writing simulation scripts.

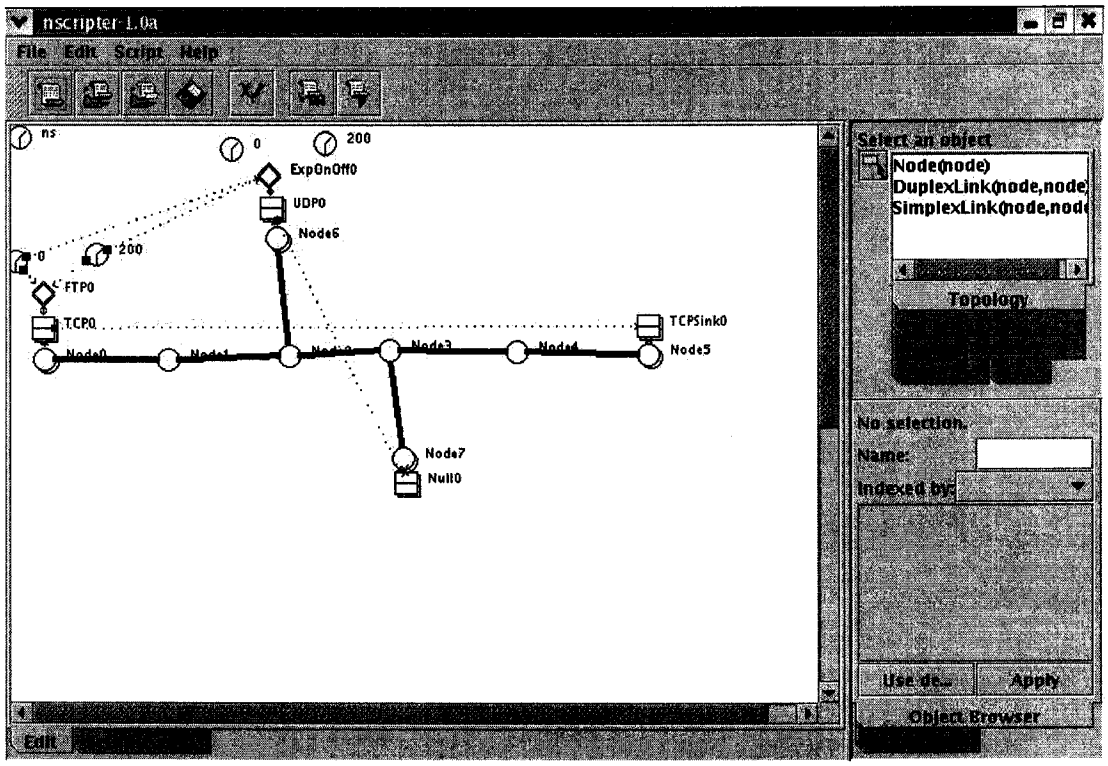


Figure 5-2: Screen shot of topology generator (Nscript)

With this kind of set up and fleet of tools we tried to observe the behavior of ACC – ABCD in various scenarios. As discussed in chapter 4, two basic scenarios of responsive and non-responsive cross traffic would vindicate our claims regarding

ACC – ABCD. In the following section we will discuss different experiments we have conducted.

5.1 Verification of ABCD Methodology

Appendix D provides details about the verification of Available Bandwidth calculation formulae. Our next step was to verify whether packet size and available bandwidth scale as we expected.

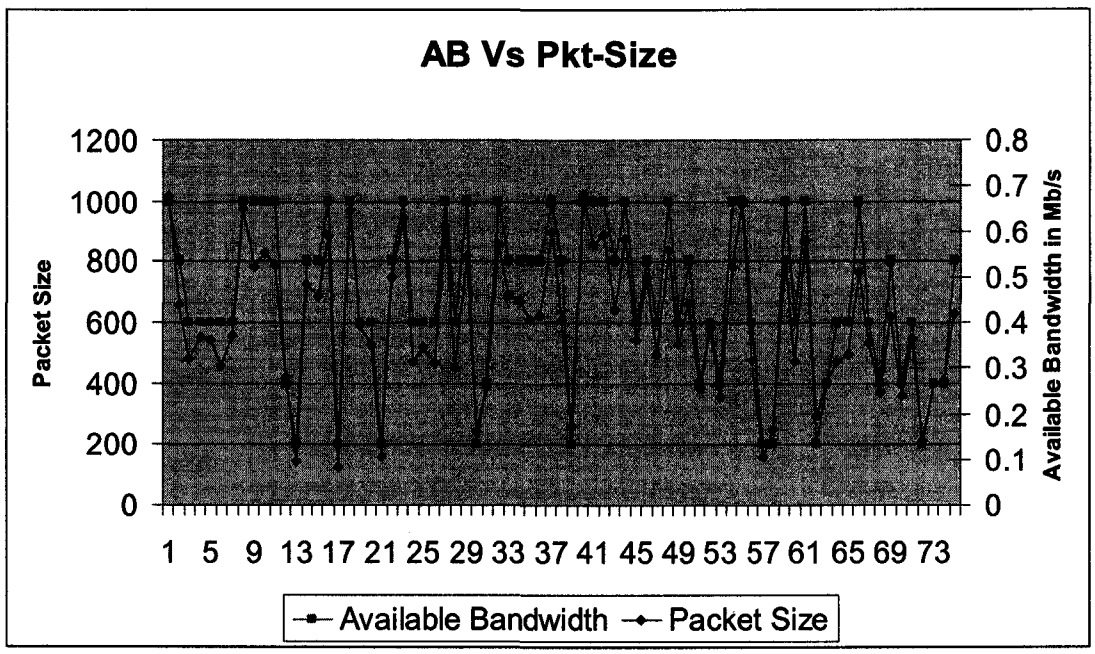


Figure 5-3: Available bandwidth Vs Packet size

Here we have shown a behavior of AB versus packet size for a period of time in our simulation. It can be seen that packet size scales well according to thresholds determined by our methodology.

Our next experiment was to check whether window size scales well with the packet size. Theoretically the window size changes linearly with the change in packet size that is because in NS-2 window size is calculated in packets and once we change the packet size the window size will automatically change in proportion.

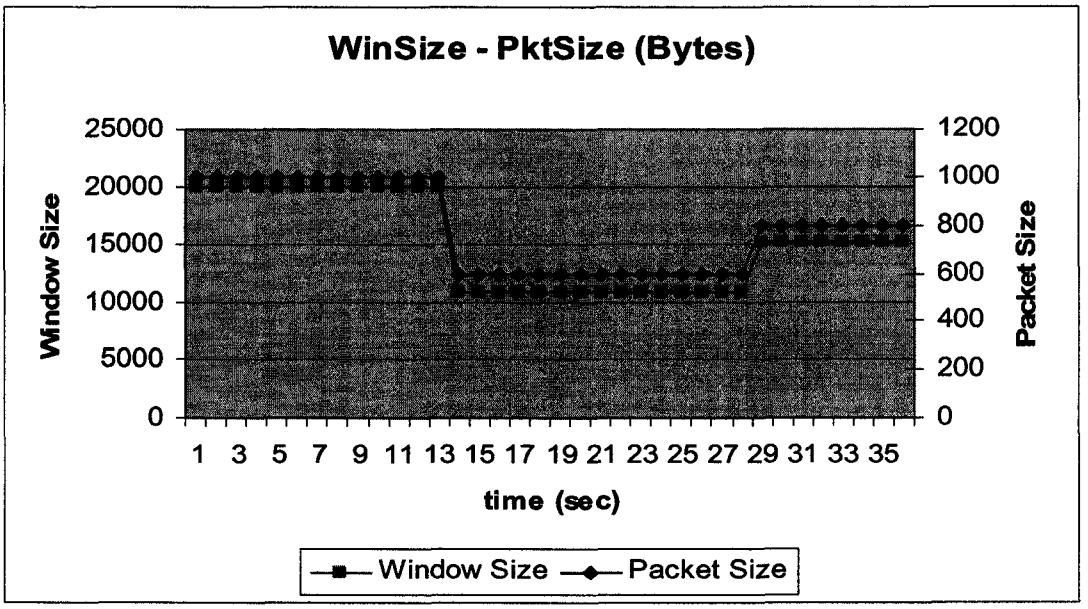


Figure 5-4: Window size - Packet size scaling with time

It can be seen from the graph that since window size is counted in packets the change in packet size reflects the same effect in window size. This behavior is observed under steady conditions but in case of a packet drop window size will scale as per the TCP algorithms. In that, case the effect will be severe.

To nullify this effect, we have to scale the window size with packet size. One more reason for doing this is, ABCD does not rely on packet drop as a congestion notification but the change in packet size signal is meant to be advance congestion notification. Hence the rate change formula discussed in chapter four was chosen.

Considering practical use of our system we did experimentation by calculating window size change in bytes. For this purpose we used rate estimation, which is mainly dependent upon packet size.

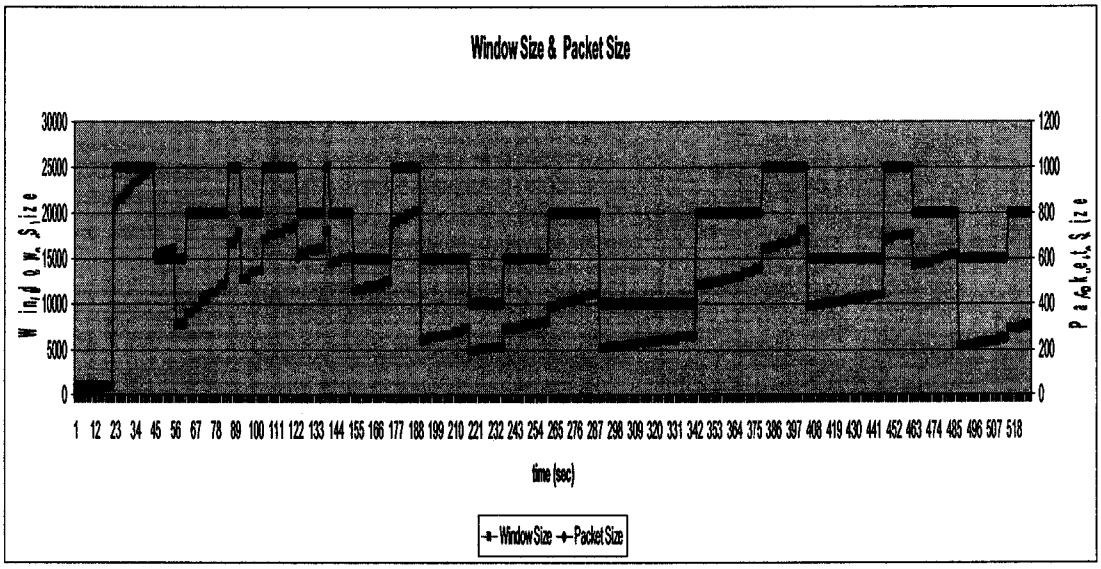


Figure 5-5: Window size and packet size

Figure 5-5 shows the behavior of window size versus packet size over a long period of time. In fig 5-6 a closer look at the trends of window size against packet size has been shown.

The gradual increase in the window size at particular moments depicts the effect of the additive increase algorithm of TCP.

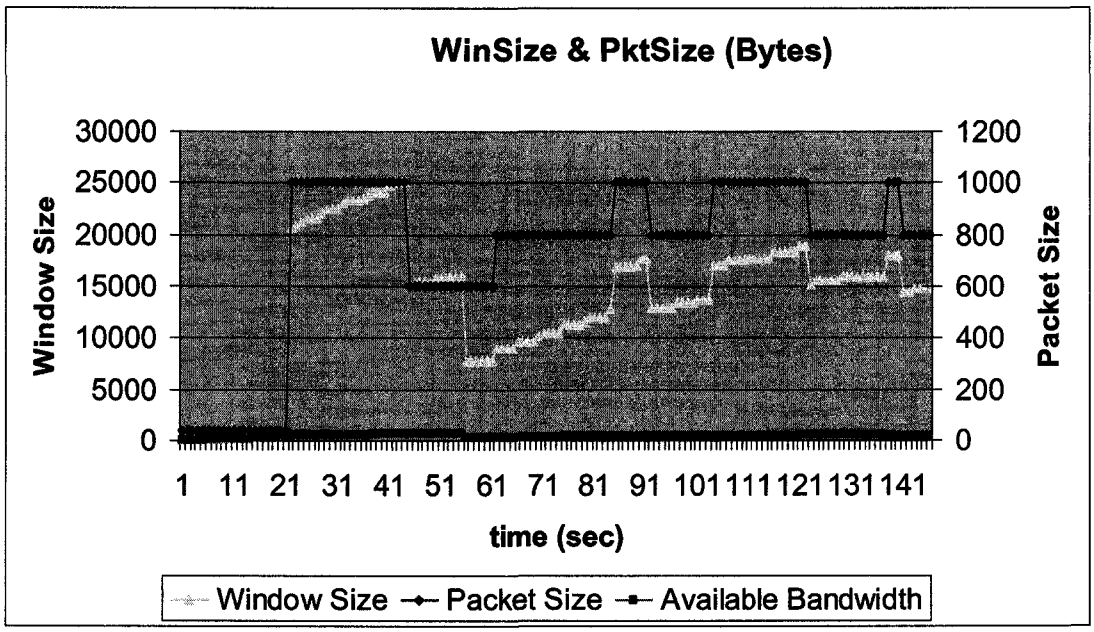


Figure 5-6: Window size and packet size

It can be observed that after halving (point 61 in figure 5-6) the window (in case of a packet drop) window size does not change with the packet size. It has to come out of the slow start process.

This means that ABCD along with TCP provides a stable environment, which will resist sudden changes (oscillations), but it will respond to increase or decrease in network capacity.

The final task was to check the behavior of window size and packet size together with the available bandwidth.

Figure 5-7 shows changes in available bandwidth, packet size and window size (in packets and bytes both) over a period of 127 seconds. To observe the changes in all the quantities we have used normalized values for generating figure 5-7. Actual values can be found in figure 5-8.

These graphs do not provide a close look at the readings at some particular points of interest. Hence in figure 5-9 we have taken readings at certain points in the simulation and it clearly shows that change in available bandwidth forces the packet size change, which is reflected in window size reduction.

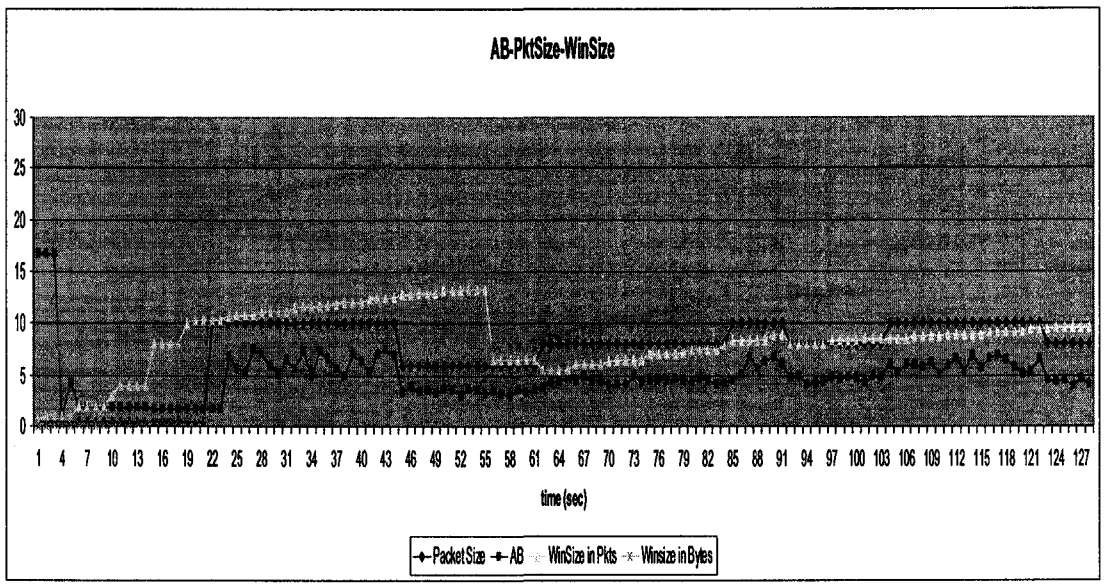


Figure 5-7: AB and Window Size (normalized)

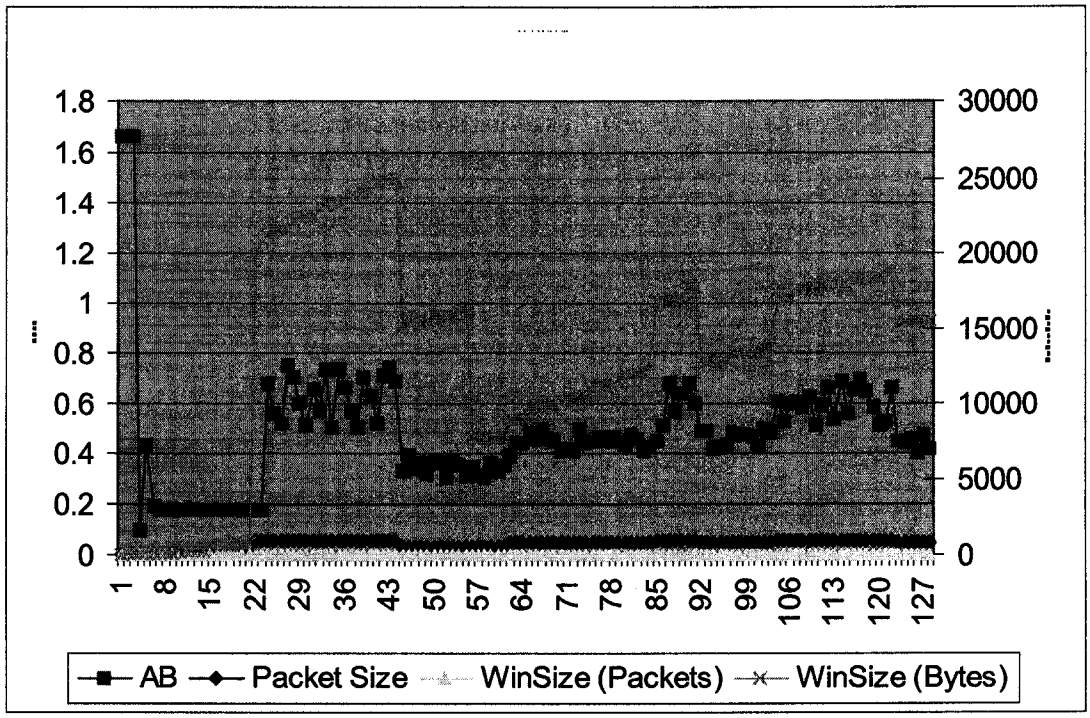


Figure 5-8: AB and Window Size

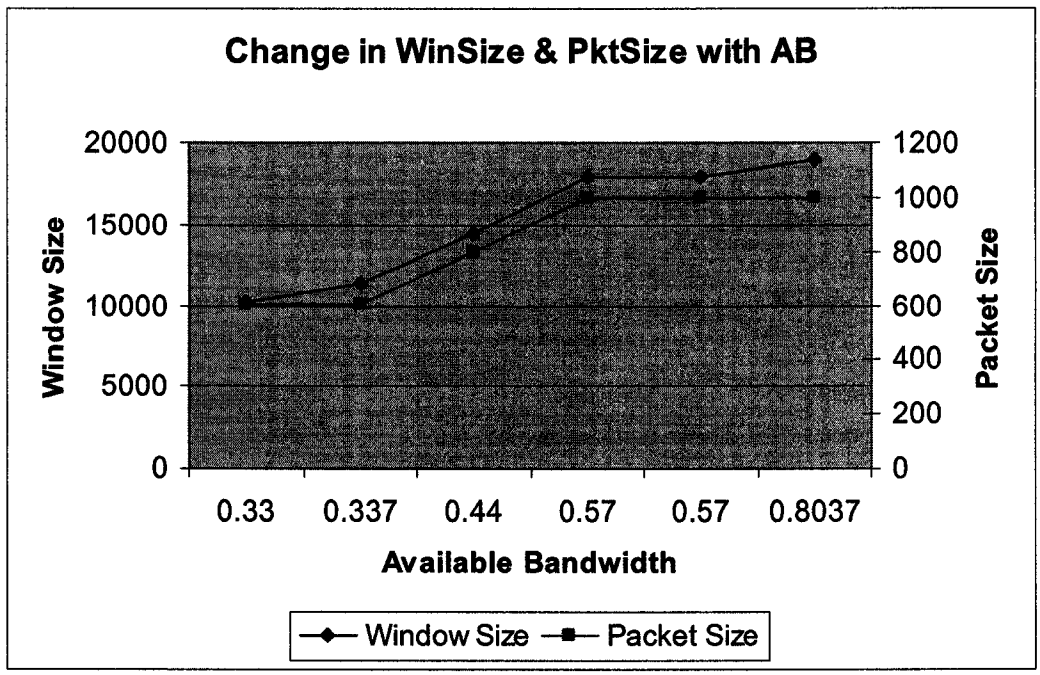


Figure 5-9: AB, packet size and window size

Hence we conclude that our theoretical claims are met by the results we get.

5.2 Simulations

The simulation experiments are run 30 times, and the data we report, is in the form of mean values over those 30 trials. To be able to see the differences among the 30 trials, we seed the simulator's number generator with the current time at each invocation.

5.3 Simulation Topology-1

We have used the well-known dumbbell topology with senders and receivers on either side of a single bottleneck link as depicted in the following figure.

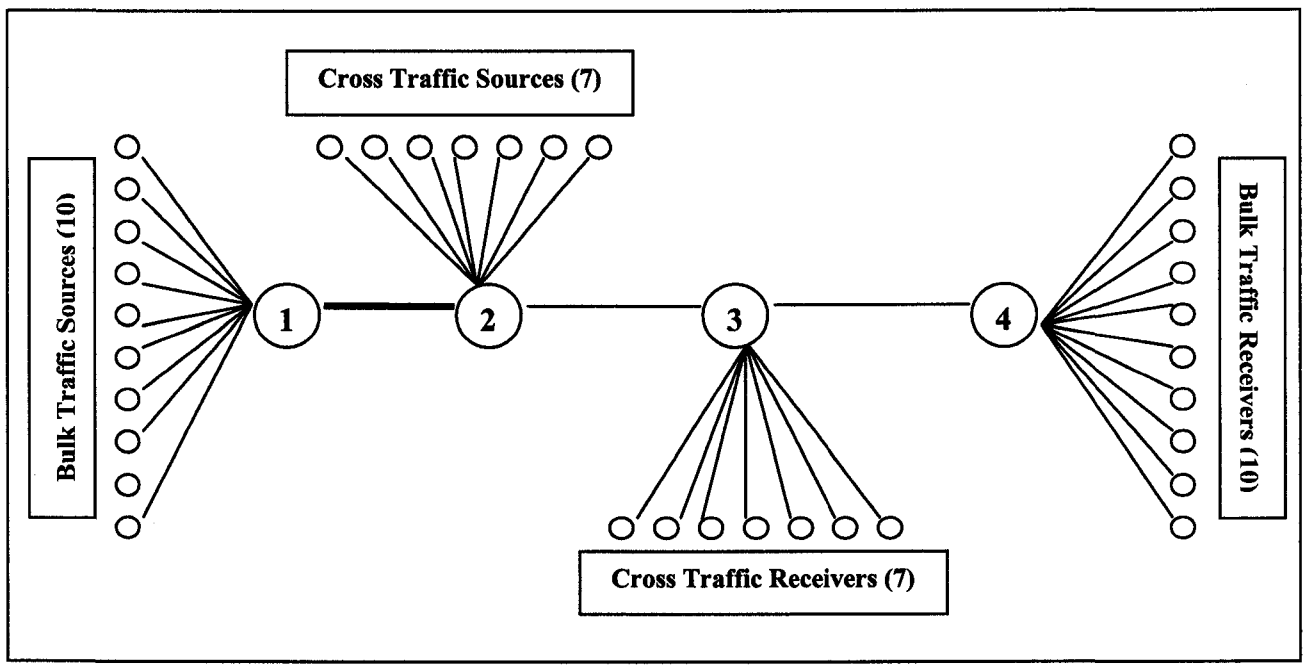


Figure 5-10: Simulation topology for ACC - ABCD

We have slightly modified the dumbbell topology by introducing link 1-2, varying the delay on this link. The bandwidth delay product that the bulk sources see is changed without changing the bandwidth-delay product that the cross traffic sees. The reason

for this modification is that it has been proven that TCP congestion control becomes unstable for large bandwidth delay products. With this kind of alteration we will be able to see the results for different bandwidth delay products.

5.4 Simulation Statistics

Consider the network and traffic configuration as follows –

- The bulk traffic sources send data continuously throughout the simulation.
- All links from an endpoint to a router have a delay of 10 ms and bandwidth of 10Mb/s except the bottleneck link, which is of bandwidth 2 Mb/s and a delay of 10 ms.
- All endpoints use 1000 byte packets.
- The cross traffic endpoints are uncontrolled endpoints, sending on-off traffic with a burst and idle time of 2.5 sec and sending rate of 100 kb/s.
- Varying the delay on the link from router 1 to 2 changes the bandwidth delay product that the bulk sources see. The simulations vary the delay between 100 and 300 ms; this range was selected because it is the high end of common round trip time in Internet.
- Each simulation lasts for 200 seconds.

	TCP	ACC	ABCD
Number of Dropped Pkts.	**886**	**716**	**24**
Throughput of Flow 1	150.03 Kb/s	150.553 Kb/s	223.886 Kb/s
Throughput of Flow 2	149.258 Kb/s	139.952 Kb/s	136.566 Kb/s
Throughput of Flow 3	158.602 Kb/s	154.455 Kb/s	135.052 Kb/s
Throughput of Flow 4	151.898 Kb/s	159.37 Kb/s	136.708 Kb/s
Throughput of Flow 5	152.345 Kb/s	172.455 Kb/s	137.133 Kb/s
Throughput of Flow 6	154.011 Kb/s	136.664 Kb/s	141.039 Kb/s
Throughput of Flow 7	158.358 Kb/s	165.67 Kb/s	134.342 Kb/s
Throughput of Flow 8	149.055 Kb/s	152.384 Kb/s	141.231 Kb/s
Throughput of Flow 9	149.014 Kb/s	151.369 Kb/s	128.492 Kb/s
Throughput of Flow 10	152.752 Kb/s	142.345 Kb/s	126.808 Kb/s
Total Throughput	**1525.32 Kb/s**	**1525.22 Kb/s**	**1441.26 Kb/s**

Table 5-1: Delay 100 ms

Table 5-1 shows the comparison of normal TCP congestion control, Faber's ACC and our ACC – ABCD system. When the delay bandwidth product is not large, Faber's ACC works comparable with TCP and gives almost the same throughput, but the loss rate is very high. Using ABCD the drop rate falls 36.9 times.

Since the packet size of retransmitted packets is changed, the throughput is reduced by 5.5%. In all the runs, the numbers of packets dropped were observed to be low. One more factor affecting the throughput is the number of lost packets. These are the packets, which were generated by sender but never reached up to the receiver. Figure 5-12 shows a simulation summary where this factor of lost packets can be found.

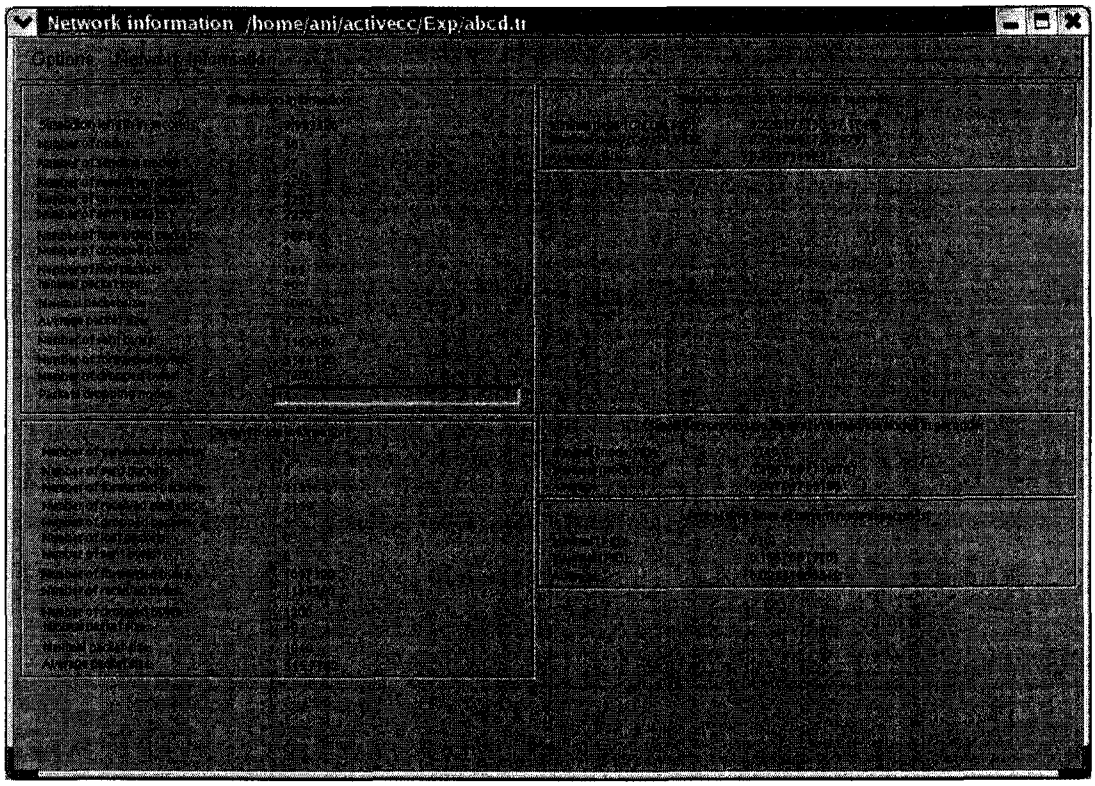


Figure 5-11: Simulation Statistics

In the figure 5-11 over all simulation summary and statistics of ABCD node have been presented. Processing times at the ABCD node sheds light on the processing overhead aspect as well. We address the overheads issue in chapter 6.

Table 5-2 shows the results for runs with the same topology as the earlier experiment but with the delay on the variable-delay link as 200 ms. As the delay bandwidth product has increased ACC's gain in throughput and ABCD's gain in (compared to the values for the 100 ms delay case) loss rate and throughput can be observed.

	TCP	ACC	ABCD
Number of Dropped Pkts.	**499**	**339**	**3**
Throughput of Flow 1	147.064 Kb/s	155.352 Kb/s	182.245 Kb/s
Throughput of Flow 2	125.736 Kb/s	153.727 Kb/s	137 Kb/s
Throughput of Flow 3	153.605 Kb/s	134.308 Kb/s	132.116 Kb/s
Throughput of Flow 4	119.358 Kb/s	131.586 Kb/s	119.923 Kb/s
Throughput of Flow 5	132.764 Kb/s	144.342 Kb/s	133.252 Kb/s
Throughput of Flow 6	122.567 Kb/s	152.102 Kb/s	122.05 Kb/s
Throughput of Flow 7	139.427 Kb/s	158.358 Kb/s	130.478 Kb/s
Throughput of Flow 8	147.186 Kb/s	133.698 Kb/s	121.213 Kb/s
Throughput of Flow 9	142.23 Kb/s	146.698 Kb/s	117.138 Kb/s
Throughput of Flow 10	141.214 Kb/s	154.661 Kb/s	120.519 Kb/s
Total Throughput	**1371.15 Kb/s**	**1464.83 Kb/s**	**1315.93 Kb/s**

Table 5-2: Delay 200 ms

In our simulations, it would be of interest to observe the delays that incur on the congested link. Figure 5-12 gives the frequency jitter of delay between node 11 and node 12.

Figure 5-11 shows that packets are dropped at node 11. Figure 5-13 represents through a 3-D graph, the number of dropped packet at various nodes. It can again be observed that all the packet drops are at the node 11.

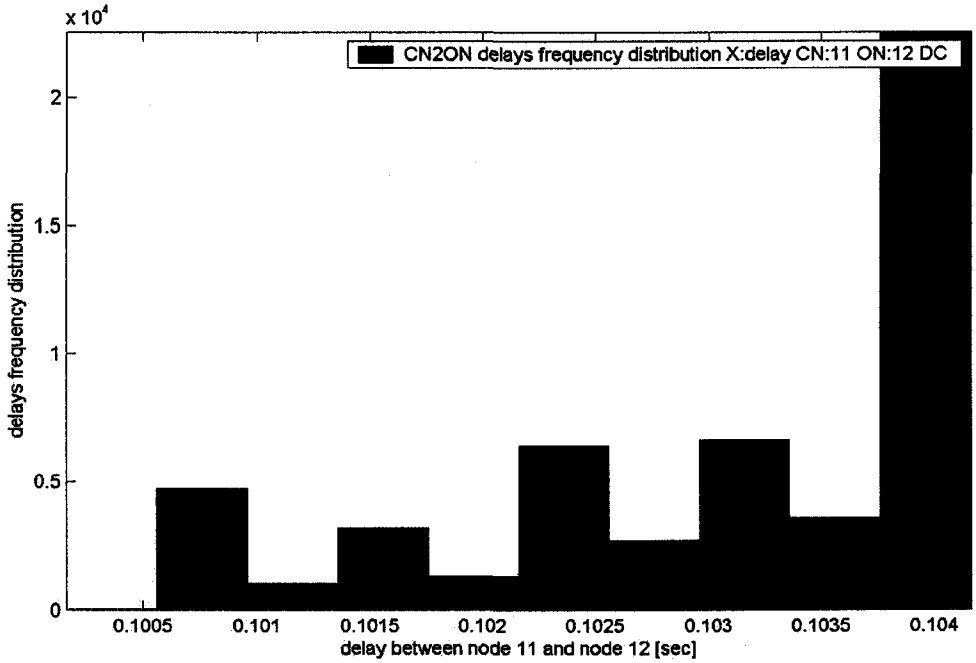


Figure 5-12: Frequency distribution

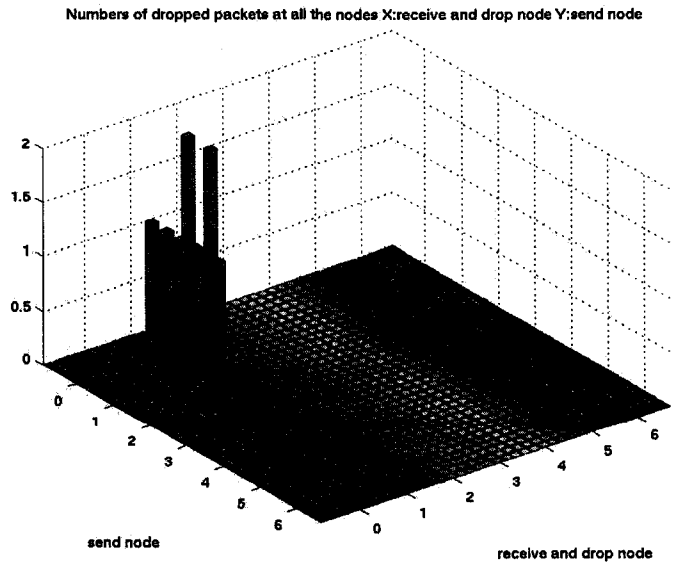


Figure 5-13: Number of dropped packets

In ABCD the throughput of first flow is found to be high. We observed that packets of this flow are mostly the first ones to reach the bottleneck (start of the flow is randomized still this flow was always getting the preference ahead of other flows) and they utilize it for first few seconds when there is no other traffic. That is why their throughput first goes high and then becomes steady. This can be observed in figure 5-11.

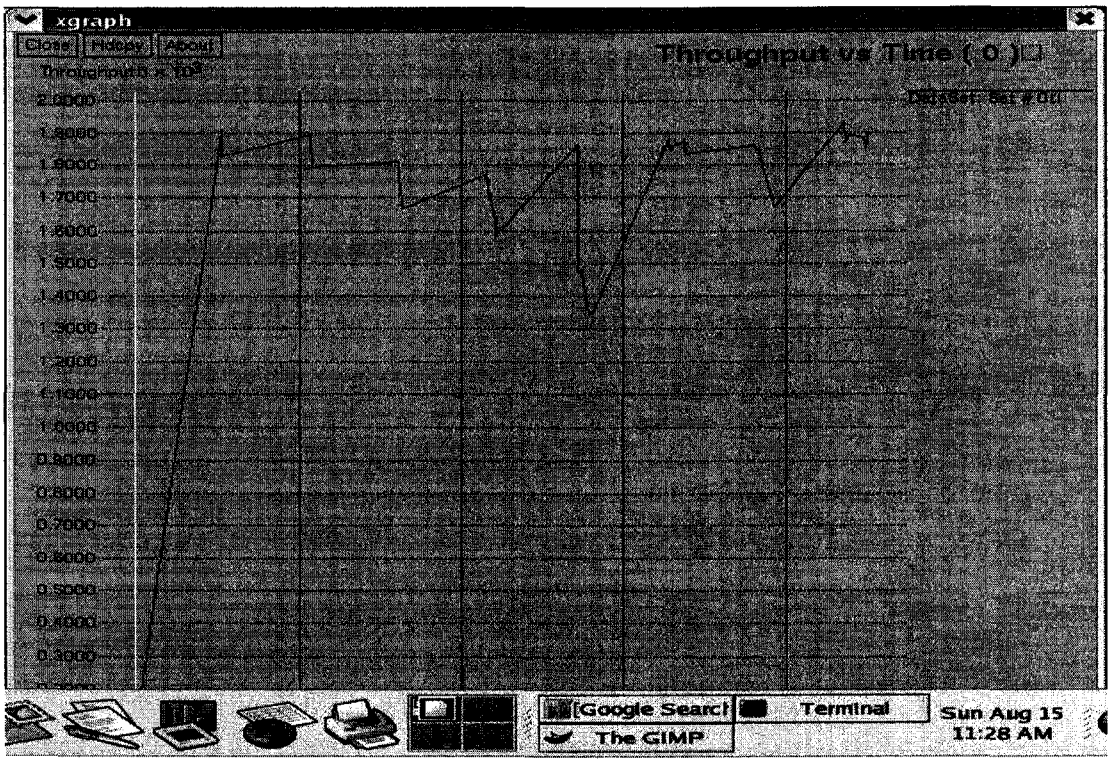


Figure 5-14: Throughput of flow 1

	TCP	ACC	ABCD
Number of Dropped Pkts.	344	161	0
Throughput of Flow 1	122.689 Kb/s	136.664 Kb/s	177.858 Kb/s
Throughput of Flow 2	136.989 Kb/s	143.245 Kb/s	130.38 Kb/s
Throughput of Flow 3	134.023 Kb/s	144.505 Kb/s	126.741 Kb/s
Throughput of Flow 4	135.364 Kb/s	130.936 Kb/s	128.22 Kb/s
Throughput of Flow 5	134.389 Kb/s	160.673 Kb/s	127.83 Kb/s
Throughput of Flow 6	130.286 Kb/s	143.205 Kb/s	128.084 Kb/s
Throughput of Flow 7	120.577 Kb/s	132.033 Kb/s	124.542 Kb/s
Throughput of Flow 8	114.93 Kb/s	136.542 Kb/s	119.555 Kb/s
Throughput of Flow 9	121.186 Kb/s	145.927 Kb/s	128.503 Kb/s
Throughput of Flow 10	132.764 Kb/s	144.18 Kb/s	127.856 Kb/s
Total Throughput	1283.2 Kb/s	1417.91 Kb/s	1319.57 Kb/s

Table 5-3: Delay 300 ms

With the high bandwidth delay product the performance of TCP is deteriorating and the improved performance of ABCD as compared to TCP, both in loss rate and throughput can be observed.

With a varied packet size marginal improvement in throughput is also considered significant because TCP is considered as biased with packet size. However with the use of ABCD, the bias of TCP in favor of larger packets is moderated. To show this, we compute the fairness index for the ten flows in our simulation.

The throughput curve of each flow varies with time. This is because the bandwidth share that each flow gets and each flow's packet size go on varying. But it can be observed from the following set of curves that each flow gets a fair share of resources and the average throughput is always comparable. Although ABCD changes the packet size at random (depending upon which packet is in the queue), this action is applied, to all flows in turn.

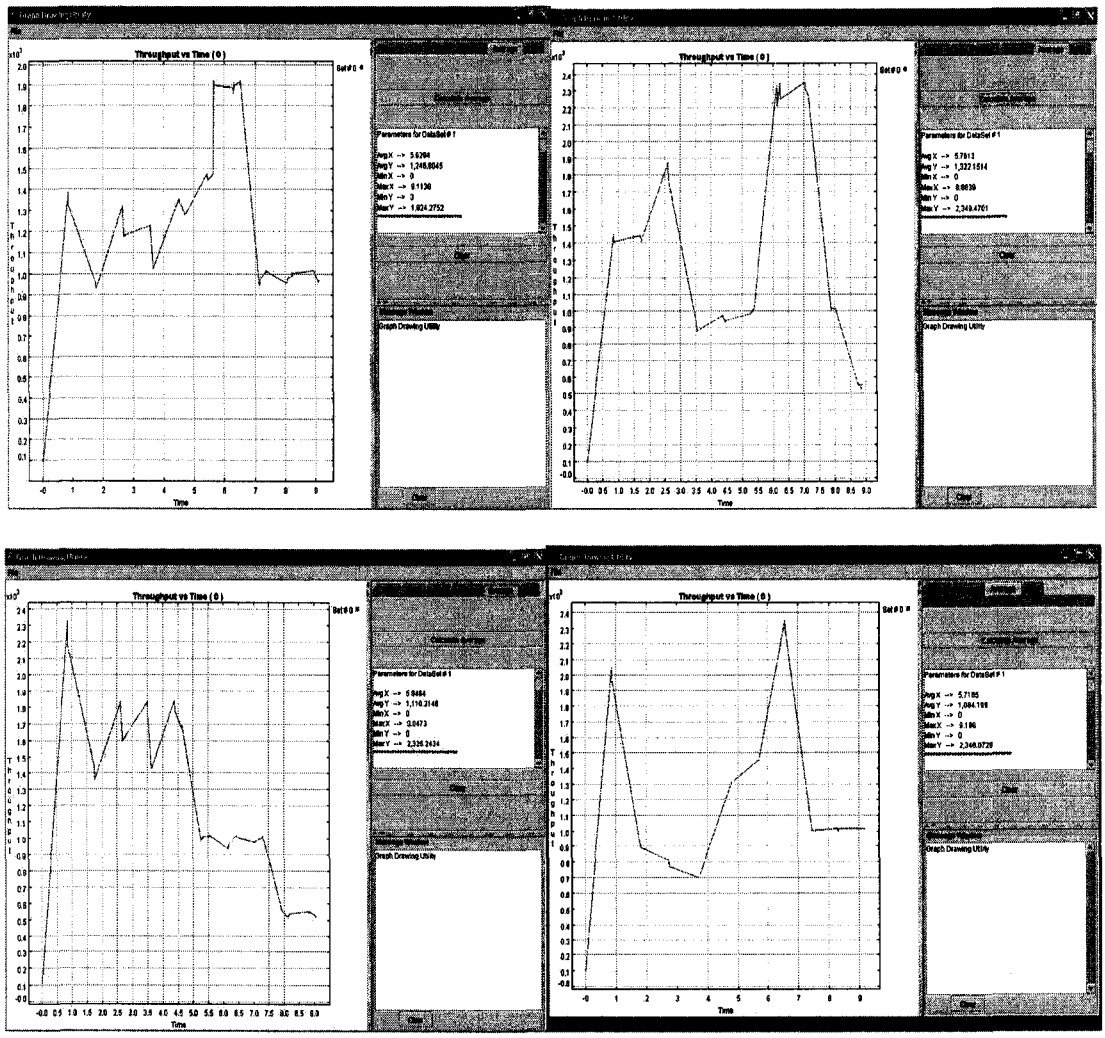


Figure 5-15: Difference in curves of throughput

5.5 Some Observations

Processing time at the node where packet size has been changed is of more interest, since it is expected that the introduction of packet size modulation will increase the requirement for processing at intermediate node.

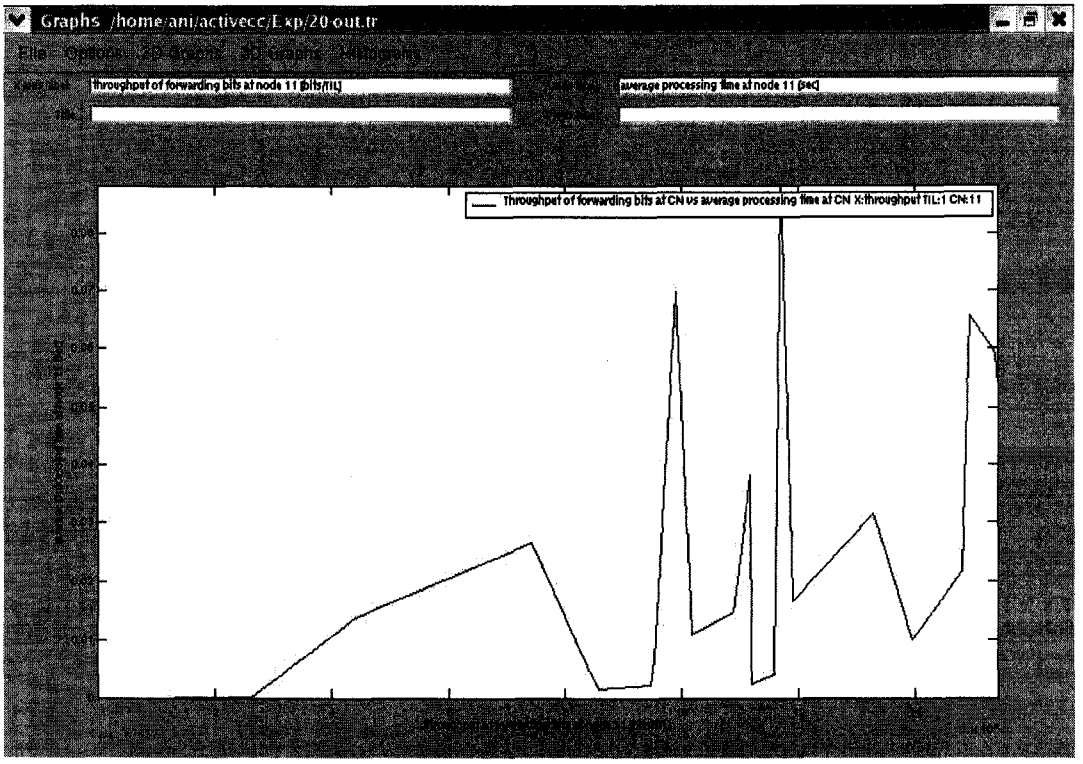


Figure 5-16: Forwarding bits Vs Processing time

Average processing time depends on how many bits it is serializing. The above graph shows drastic changes in processing time.

The following two curves (figure 5-16 and 5-17) show average processing time and end-to-end delay according to packet size.

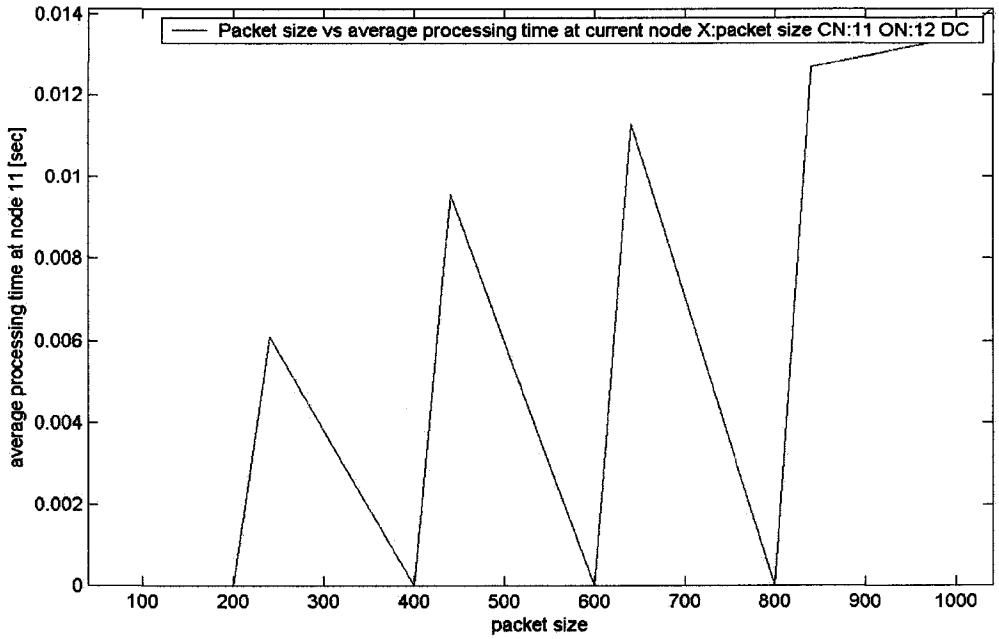


Figure 5-17: Throughput Vs average processing time

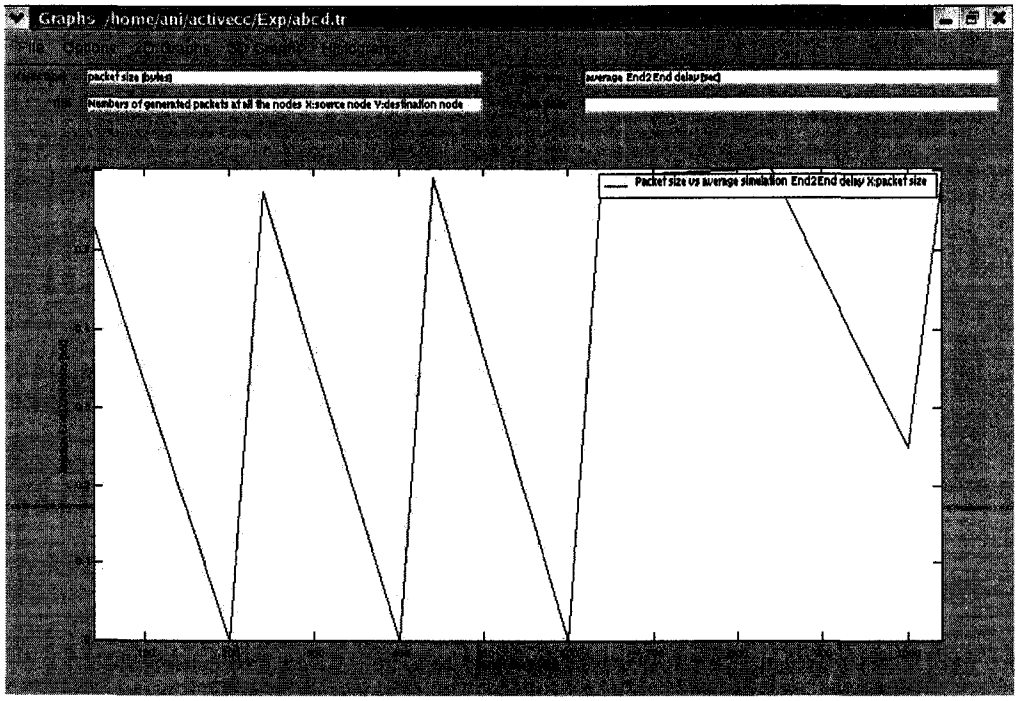


Figure 5-18: X – Axis: Pkt size (bytes) Vs Y – Axis : end to end delay (sec)

We have already seen that the packet size used by each flow need not be the same all the time. Figure 5-19 shows again that 2 different flows might generate different numbers of particular size packets. Earlier (figure 5-14) we have observed that their throughput curves though different yield almost the same throughput –

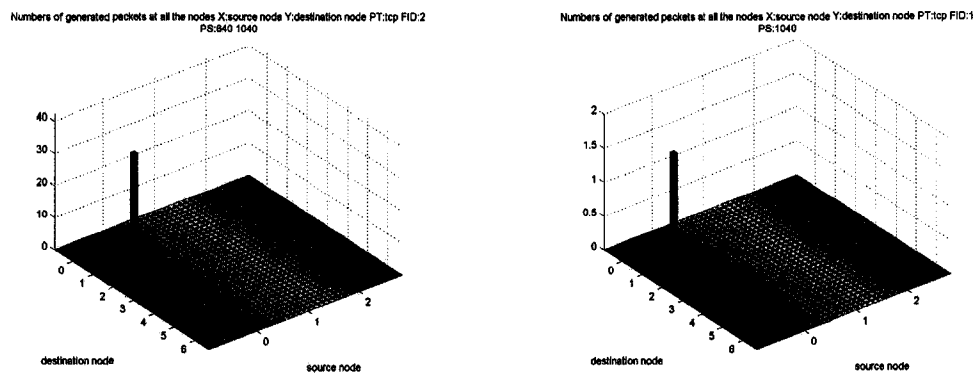


Figure 5-19: Packet generation of different size

The following curve shows throughput against packet size of several flows together -

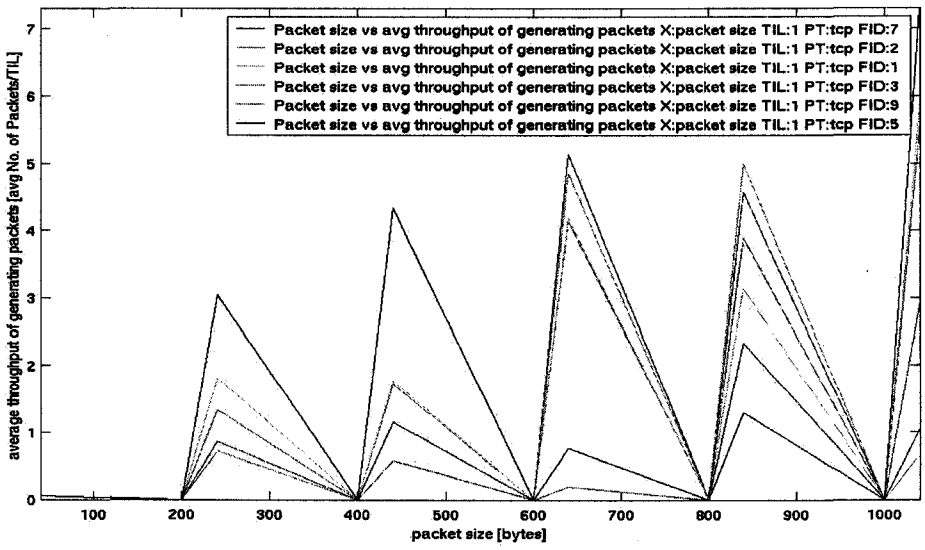


Figure 5-20 Difference throughputs

5.6 Simulation Topology-2

We consider the same dumbbell topology as before but we change the cross traffic The bandwidth of the bottleneck link is modified to 1.5 Mb/s. Instead of non-responsive (to congestion) exponential traffic, we introduced TCP flow competing with bulk traffic, which is under consideration. Here we apply ABCD to cross traffic also.

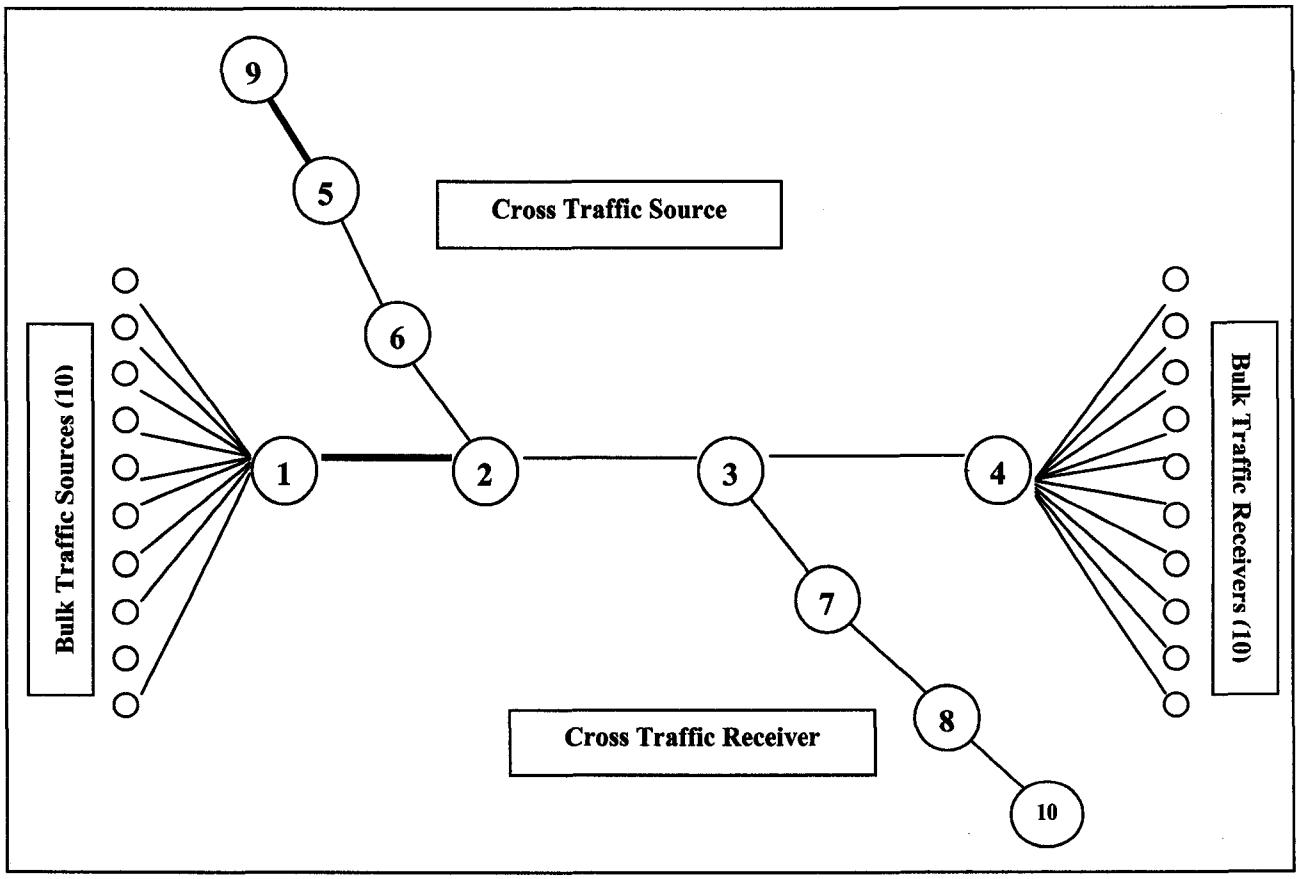


Figure 5-21: Simulation topology 2

Since we observed that at high bandwidth delay product ABCD outperforms TCP we will consider topology-2 readings for high bandwidth delay product case only.

	TCP	ABCD
Number of Dropped Pkts.	201	24
Throughput of Flow 1	151.736 Kb/s	179.702 Kb/s
Throughput of Flow 2	171.683 Kb/s	166.905 Kb/s
Throughput of Flow 3	155.067 Kb/s	168.564 Kb/s
Throughput of Flow 4	162.258 Kb/s	176.484 Kb/s
Throughput of Flow 5	136.786 Kb/s	166.014 Kb/s
Throughput of Flow 6	168.677 Kb/s	170.072 Kb/s
Throughput of Flow 7	161.12 Kb/s	167.761 Kb/s
Throughput of Flow 8	147.389 Kb/s	168.275 Kb/s
Throughput of Flow 9	163.111 Kb/s	175.939 Kb/s
Throughput of Flow 10	163.68 Kb/s	170.838 Kb/s
Total Throughput	**1581.51 Kb/s**	**1710.55 Kb/s**

Table 5-4: Topology-2 - Delay 200 ms

In this second simulation topology we observe that drop ratio at the cross traffic are almost eliminated. Traffic entering the main flow can be tuned more if available bandwidth for link 6-2 can also be observed and utilized by using ABCD method. The limitations of our processing power restricted us from implementing the ABCD in each node. But as we said earlier, we envision a system where all the intermediate nodes would be programmed with ABCD, so that whenever the available bandwidth falls, the necessary action may be taken.

	TCP	ABCD
Number of Dropped Pkts.	38	5
Throughput of Flow 1	141.58 Kb/s	149.095 Kb/s
Throughput of Flow 2	141.295 Kb/s	149.095 Kb/s
Throughput of Flow 3	140.889 Kb/s	149.095 Kb/s
Throughput of Flow 4	140.808 Kb/s	149.095 Kb/s
Throughput of Flow 5	141.377 Kb/s	149.095 Kb/s
Throughput of Flow 6	132.398 Kb/s	143.57 Kb/s
Throughput of Flow 7	140.645 Kb/s	143.652 Kb/s
Throughput of Flow 8	139.833 Kb/s	143.327 Kb/s
Throughput of Flow 9	138.452 Kb/s	143.408 Kb/s
Throughput of Flow 10	133.617 Kb/s	143.489 Kb/s
Total Throughput	1390.89 Kb/s	1462.92 Kb/s

Table 5-5: Topology-2 - Delay 300 ms

The throughputs of this topology are fairer than the earlier one. In earlier observations though the variance factor in throughputs was significant the fairness index was always steady.

5.7 Fairness Index

Since the flows are using packets of different size the fairness index of the network needs to be observed. n users sharing a path and each demanding infinite resources should have equal throughputs. If throughputs are not exactly equal, the fairness function can be quantified using the following fairness function -

$$\text{Fairness } (x_1, x_2 \ldots x_n) = \frac{\left(\sum_{i=i}^{n} xi\right)^2}{n.\sum_{i=1}^{n} x^2 i}$$

where x = i th users throughput

If all the users do not get exactly equal allocations in a system, the system is less fair and we need an index function.

The fairness index returned by this function is bounded between 0 and 1. A totally fair allocation has a fairness index of 1. This function is independent of scale, i.e., unit of measurement does not matter.

For the throughputs we measured with ABCD the fairness index, is approximately equal to 0.99. With the variable packet size TCP flows the unfairness is negligible. TCP is always considered as unfair when there are variable packet size flows. For TCP the difference in throughput is of the order of difference in packet size.

In our earlier analysis of results we observed that packet size in ABCD ranges from 200 to 1000 and a fairness index close to 1 supports our claim that with ABCD system resources are shared fairly.

5.8 Results at a glance

	TCP			ACC			ABCD		
	T	D	F	T	D	F	T	D	F
100 ms	1525.32 Kbps	886	0.999502	1525.22 Kbps	716	0.995131	1441.26 kbps	24	0.96622
200 ms	1371.15 Kbps	499	0.993677	1464.83 Kbps	339	0.99578	1315.93 Kbps	3	0.988414
300 ms	1283.2 Kbps	344	0.996738	1417.91 Kbps	161	0.996739	1319.57 Kbps	0	0.986293

Table 5-6: Topology-1 Summary

T = Throughput, **D** = Number of dropped packets, **F** = fairness Index

	TCP			ABCD		
	T	D	F	T	D	F
100 ms	1581.51 Kbps	201	0.996031	1710.55 Kbps	24	0.999324
200 ms	1390.89 Kbps	38	0.999475	1462.92 Kbps	5	0.999631

Table 5-7: Topology-2 Summary

T = Throughput, **D** = Number of dropped packets, **F** = fairness Index

5.9 Summary

In this chapter we analyzed the results of our ACC - ABCD methodology. Our investigations show that our goal of having a fair system with variable packet size with decreased loss and drop rate have been achieved. In the last chapter we will summarize the thesis and outline the future work that needs to be done.

Chapter 6 . Conclusions and Future Work

"Any improvement in performance offered to one service class will be at the expense of a reduction in the service levels offered to other service categories. You can't make a perpetual motion machine. You can't make photons travel faster than the speed of light. But you can create networks active or passive that unilaterally improve the delay and loss characteristics of all packets passed through a network." – [IETF, IPMM Mail archives, February 1999]

6.0 Conclusions

For congestion control, this thesis proposed the method of limiting the number of bytes entering the network when the bottleneck resource is bandwidth. The existing approaches of TCP congestion control and Active Congestion Control were revisited, re-evaluated, and the idea was extended to reflect the effect of Available Bandwidth.

It was shown how a packet size adjustment capability mitigates the situation of congestion in the Internet. Flow rate adjustment at the sender in addition to packet

size adjustment at intermediate node, improves the throughput even in high bandwidth delay product case.

We envision a system where, if all the intermediate nodes are ABCD compliant, each flow will be fair to another. No flow will encroach onto another flow's resources. As results in Chapter 5 show the processing overheads are increased with the introduction of computation at the node. Hence for improving the practicability of Active Networks, the processing power at each of the intermediate nodes will have to be increased substantially. However a quantitative study of the overhead may require the use of a tool, which is more sophisticated than NS-2.

Over 90% of the Internet traffic is TCP. If such a large amount of Internet traffic can be controlled, this will improve the overall performance of Internet. The argument against variance in packet size might be that it introduces overheads but studies over the MCI backbone 2002 show that most TCP flows are less than 30 packets long.

6.1 Questions that need to be answered

The following questions have been raised since the emergence of active networks. The Active Network community identifies them as challenges and understands that for real world use of active networks will need answers to these questions.

- **Processing overhead** — How can programs in packets be installed and run efficiently?
- **Network overhead** — How do we keep programs small?

- **Security versus flexibility** — Who is allowed to run what, and how do you enforce it?
- **Heterogeneity** — What languages, resources and local services does a node support?
- **Acceptance** — Who will expose their hardware to programmability?

The argument in favor of Active Network was that this kind of programability equips networks to handle many clumsy problems (such as congestion) in a non-traditional but efficient way. Hence there can be some trade-off with efficiency of the overall network.

6.2 What needs to be done? (Future Work)

All the projects that were part of *'Abone'* were regarded as a practically infeasible. But with the successful implementation of 'Active-IP' [Wetherall 96], the research community started regarding this area as a potential research topic.

With the possibility of cross-over from IP version 4 to IP version 6 the ABCD technique will be obsolete, since there will be no longer a fragmentation capability in routers. IP version 4 may remain as an integral part of Internet for many years.

But in the long run the issue of Active Networks will have to be addressed. We think that it needs to be seen what difference would it make to the performance indices, if end-to-end available bandwidth is probed (path MTU kind of techniques will be useful) and action would be taken at the sender only.

Again networks will be facing the problem of feedback delay, but 'report-n-route' kind of probing techniques (as discussed in chapter 3) can be used for this purpose.

From the security point of view the problems like 'Syn-flooding' can be solved using this technique. Incipient malicious flood will be detected as the available bandwidth goes down and irrespective of whether it is a attack or routine traffic, it will be edited before it makes the bottleneck link situation chaotic.

As a next step, like any other active network application our goal is to try it out on a real testbed. Shortcomings will be highlighted with actual use and those aspects can be revisited with the Simulator.

Bibliography

[Aggarwal 04] A. K. Aggarwal, A. N. Bharadwaj; ***'ACC-ABCD compliant NS-2'***, School of Computer Science, University of Windsor, Canada, Technical Report number UofWSCS04-020, 2004.

[Banks] **'Discrete-event System Simulation'**, by J. Banks et al. Prentice Hall (1996), ISBN 0 -13 -217449-9

[Bhattacharjee 96] S. Bhattacharjee, K. Calvert, and E. Zegura; ***'On Active Networking and congestion'*** College of Computing, Georgia Tech, Tech. Rep. GITCC -96-02, 1996.

[Bhattacharjee 97] Samrat Bhattacharjee, Ken Calvert, and Ellen W. Zegura.; ***'Active Networking and the End-to-End Argument'*** IEEE International Conference on Network Protocols (ICNP '97), pp. 220 – 229, Atlanta, GA, October 28-31, 1997.

[Brakmo 95] L. Brakmo and L. Peterson.; ***'TCP Vegas: End to End Congestion Avoidance on a Global Internet'***, IEEE Journal of Selected Areas in Communications, 13(8):1465 1480, October 1995.

[Calvert 98] K.L. Calvert, S. Bhattacharjee, E. W. Zegura, and J. Sterbenz; ***'Directions in active networks'***, IEEE Communications Magazine, vol. 36, no. 10, pp. 72–78, Oct. 1998.

[Campbell 99] A. Campbell, H. De Meet, M. Kounavis, K. Miki, J. Vicente, and D. Villela; ***'A Survey of Programmable Networks'***, ACM Computer Communications Review, April 1999.

[Cheng 01] Cheng Wanxiang; Shi Peixin; Lei Zhenming; ***'Network-assisted congestion control'***, Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences on, Volume: 2, pp.8 – 32, 29 Oct.-1 Nov. 2001.

[Cheng 03] Lechang Cheng; Ito, M.R.; ***'Layered multicast with TCP-friendly congestion control using active networks'***, 10th International Conference on Telecommunications, ICT 2003.Volume: 1, pp.806 – 811, 23 Feb.-1 March 2003.

[Faber 02] Faber, T; ***'Experience with active congestion control'***, DARPA Active NEtworks Conference and Exposition, 02, pp.132 - 142, 29-30 May 2002.

[Faber 98] Faber, T; ***'ACC: using active networking to enhance feedback congestion control mechanisms'***, Network, IEEE, Volume: 12 Issue: 3, pp.61 – 65, May-June 1998.

[Floyd 93] Floyd, S., and Jacobson, V.; ***'Random Early Detection gateways for Congestion Avoidance'***, IEEE/ACM Transactions on Networking, V.1 N.4, pp. 397-413, August 1993.

[Floyd 99] Floyd, S.; Fall, K.; ***'Promoting the use of end-to-end congestion control in the Internet'***, IEEE/ACM Transactions on Networking, Volume: 7, Issue: 4, pp.458 – 472, Aug. 1999.

[Golestani 98] Golestani, S.J.; Bhattacharyya, S.; ***'A class of end-to-end congestion control algorithms for the Internet'***, Sixth International Conference on Network Protocols, pp.137 –

150, 13-16 Oct. 1998.

[Gyires 00] Gyires, T.; ***'Using active networking for congestion control in high-speed networks with self-similar traffic'***, IEEE International Conference on Systems, Man, and Cybernetics, Volume: 1, pp.405 – 410 8-11 Oct. 2000.

[Huang 02] Hsuan Huang; ***'Active Networks: An Overview'***, Technical report – Department of computer engineering and science, Yuan Ze University, 2002.

[Jacobson 88] Jacobson V.; ***'Congestion Avoidance and Control'***, Computer Communication Review, vol. 18, no.4, Pages: 314-329 ,August 1988

[Jain 88] Jain R, Ramakrishnan K, Chiu D.; ***'Congestion avoidance in computer networks with a connectionless network layer'***, Proceedings of the Computer Networking Symposium, pp.134 – 143, 11-13 April 1988.

[Janarthanan 03] Yoganandhini Janarthanan,Gary Minden, and Joseph Evans; ***'Enhancement of Feedback Congestion Control Mechanisms by Deploying Active Congestion Control,*** The University of Kansas, Information and Telecommunication Technology Center, Technical report, ITTC-FY2003-TR-19740-10.

[Johari 01] Johari, R.; Tan, D.K.H.; ***'End-to-end congestion control for the Internet: delays and stability'***, IEEE/ACM Transactions on Networking, Volume: 9, Issue: 6, pp.818 – 832, Dec. 2001.

[Katabi 02] Dina Katabi, Mark Handley, Charlie Rohrs; ***'Congestion control for high bandwidth-delay product networks,*** ACM SIGCOMM Computer Communication Review , Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, Volume 31 Issue 4, 2002.

[Kawamura 00] Kawamura, R.; Stadler, R.; ***'Active distributed management for IP networks',*** Communications Magazine, IEEE, Volume 38, Issue: 4, pp.114 – 120, April 2000.

[Lau 01] W. Lau, S. Jha, and M. Hassan; ***'Current directions in Active Programmable Network'*** IEEE International Conference on Networks ICON, Bangkok, October 2001.

[Lemar 99] Eric Lemar, Stefan Bjarni Sigurosson; ***'Congestion Control in Active Networks'***, Technical Report, University of Washington 1999.

[Li 03] Jiang Li, Shivkumar Kalyanaraman; **'MCA: A Rate-based End-to-end Multicast Congestion Avoidance Scheme',** Communications, 2002. ICC 2002. IEEE International conference on , Volume: 4 , pp. Pages:2341–2347,28 April-2 May 2002.

[Lotfi 93] B Lotfi, Semyon M. Meerkov; ***'Feedback control of congestion in packet switching networks: the case of a single congested node',*** IEEE/ACM Transactions on Networking (TON), V-1/6, pp.693 – 708, December 1993.

[Mo 98] Mo, J. and Walrand, J.; ***'Fair End-to-End Window-based Congestion Control'***, IEEE/ACM Trans. on Networking 8-5, pp. 556-567, 2000.

[Psounis 99] Konstantinos Psounis; ***'Active Networks: Applications, Security, Safety and Architectures'***, IEEE Communications Surveys, Vol. 2, No.1, First Quarter, 99.

[RFC 1185] V. Jacobson, R. Braden, L. Zhang ; ***'TCP Extension for High-Speed Paths'***, October 1990.

[RFC 2001] W. Stevens ; ***'TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms'***, January 1997.

[RFC 2018] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow ; ***'TCP Selective Acknowledgement Options'***, October 1996.

[RFC 2481] K. Ramkrishnan, S. Floys ; ***'A Proposal to add Explicit Congestion Notification'***, January 1999.

[RFC 2581] M. Allman, V. Paxon ; ***'TCP Congestion Control'***, April 1999.

[RFC 2883] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky; ***'An Extension to the Selective Acknowledgement (SACK) Option for TCP'***, July 2000.

[RFC 3168] K. Ramkrishnan, S. Floyd. D. Black ;***'The Addition of Explicit Congestion Notification (ECN) to IP'***, September 2001.

[RFC 3782] S. Floyd, T. Henderson, A. Gurtov; ***'The NewReno Modification to TCP's Fast Recovery Algorithm'***, April 04.

[RFC 793] V. Jacobson ; ***'Transmission Control Portocol'***, DARPA INTERNET ROGRAM PROTOCOL SPECIFICATION, September 1981.

[Stadler 02] R. Stadler; ***'Decentralized and adaptable management based on active networking technology'*** International Workshop on Active Network Technologies and Applications (ANTA 02), Tokyo, Japan, March 25-26, 2002.

[Stoica 03] I. Stoica, S. Shenker, and H. Zhang.;***'Core-Stateless Fair Queuing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks.'*** Proceedings of the SIGCOMM '98 Conference, Vancouver, Canada, Aug. 1998.

[Tennenhouse 96] David Tennenhouse, and David Wetherall; ***'Toward an Active Network Architecture'***, ACM SigComm's Communication Review, April 1996.

[Tennenhouse 97] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden; ***'A Survey of Active Network Research'***, IEEE Communications Magazine, Vol. 35, No. 1, pp80-86. January 1997.

[Wang 00] Bin Wang; Bing Zhang; Zengji Liu; Hongbin Li; ***'Forward active networks congestion control algorithm and its performance analysis'***, The 2000 IEEE Asia-Pacific Conference on Circuits and Systems, IEEE APCCAS 2000, pp.313 – 318, 4-6 Dec. 2000.

[Wetherall 96] David J. Wetherall, David L. Tennenhouse; ***'The Active IP Option'***, Proceedings of the 7th ACM SIGOPS European Workshop, Connemara, Ireland, pp. 33 – 40, Sept. 1996.

[Widmer 00] J Widmer; ***'Equation Based Congestion Control'***, Department of Mathematics and Computer Science, University of Mannheim, February 2000 .

[Widmer 01] J Widmer, R Denda, and M Mauve; ***'A Survey on TCP-Friendly Congestion Control'***, Special Issue of the IEEE Network Magazine, pp.28-37, May 2001.

[Widmer 02] J Widmer, C Boutremans, J Boudec; ***'End-to-end congestion Control for Flows with Variable Packet Size'***, Technical Report ID – IC/2002/82, University of Mannheim, Manheim, Germany. 2002.

[Williamson 98] Williamson, B.; Farrell, C.; ***'Active congestion control'***, Global Telecommunications Conference, GLOBECOM 98. The Bridge to Global Integration. IEEE, Volume: 3, pp.1509 – 1514 8-12 Nov. 1998.

[Xicheng 02] Xicheng Liu; Cheung, C.; Baras, J.S.; ***'Generic congestion control through network coordination'***, The 8th International Conference on Communication Systems, 2002. ICCS 2002. , Volume: 2, pp.903 – 907, 25-28 Nov. 2002.

Appendix: A - Glossary of Terms

ACC – Active Congestion Control

ABCD – Available Bandwidth-based Congestion Detection

AN – Active Networks

PN – Programmable Networks

APN – Active Programmable Networks

ANEP – Active network Encapsulation Protocol

E2E – End to End

ADM – Active Distributed Management

ANTS – Active Network Transport System

PAN – Practical (Capsule Based) Active Network

Appendix: B – Algorithms Used in TCP Variants

- **Slow Start**

Earlier versions of TCP start a connection with the sender injecting multiple segments into the network, up to the window size advertised by the receiver. Some intermediate routers queue the packets, and packets might be dropped (for various possible reasons) also.

Slow start algorithms determines at what speed the sender is acknowledging the packets and adjusts the sender rate accordingly. It helps the network from entering into the congestion right at the beginning of connection or right after the packet loss.

- **Congestion Avoidance**

Congestion avoidance and slow start looks to have same functionality but they are independent algorithms with different objectives. When congestion occurs TCP must slow down its transmission rate of packets into the network, and then invoke slow start to get things going again.

The grey numbered boxes are packets and white numbered boxes are corresponding acknowledgements. As and when ACK is received two packets are inserted, one for the ack (the ack says a packet has left the system so a new packet is added to take its place) and one because an ack opens the congestion window by one packet.

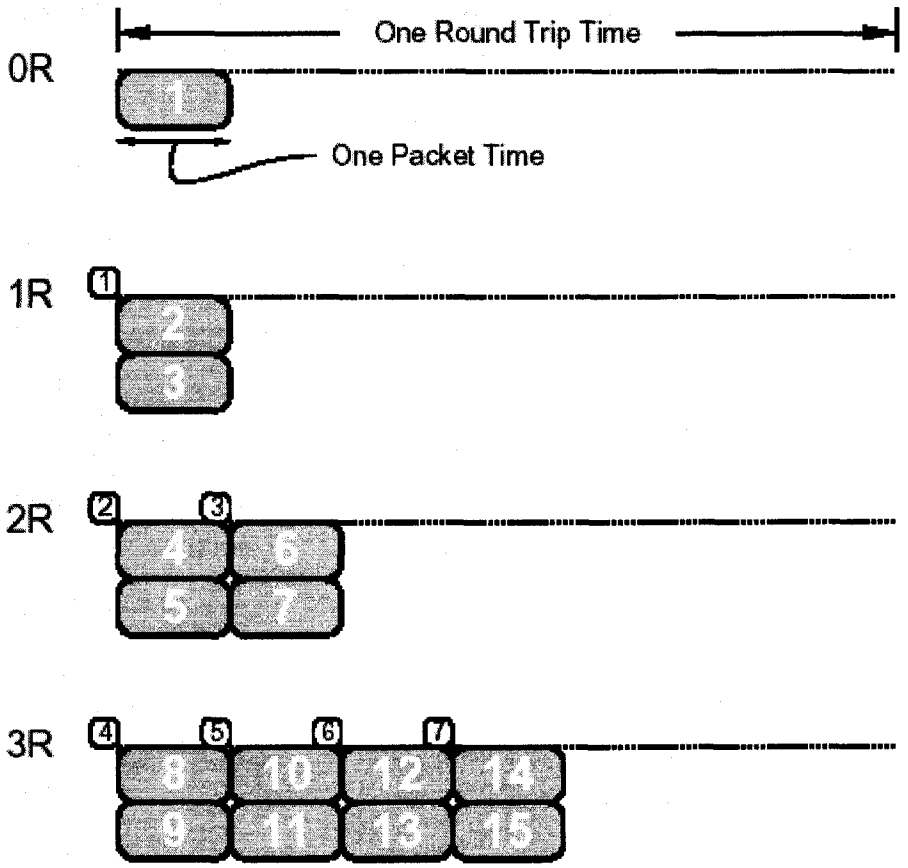


Figure B-1: Chronology of slow start *([Jacobson 98] pp.6)*

It is clear from the figure why an add-one packet-to-window policy opens the window exponentially in time. Hence slow start and congestion avoidance proves to be useful.

- **Fast Retransmit**

Congestion avoidance algorithm was modified in 1990. TCP generates an immediate acknowledgment (a duplicate ACK) when an out-of-order segment is received. This duplicate ACK serves the purpose of letting the sender know that a segment was received out of order, and to tell it what sequence number is expected.

The sender does not know the cause of duplicate ACK and waits for a couple of duplicate ACKs before retransmission because it could be a result of reordering of segments. There is a assumption that if reordered segment is received after a while then number of duplicate ACKs will not be more than three. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then retransmits the missing segment, without waiting for a retransmission timer to expire.

- **Fast Recovery**

After fast retransmit sends the missing segment, congestion avoidance is performed. This is the fast recovery algorithm. There is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start.

One more reason for not performing slow start is that the receipt of the duplicate ACKs tells TCP more than a packet has been lost. It is an improvement that allows high throughput under moderate congestion, especially for large windows.

- **Go-Back-n**

Basic idea and significance of letter 'n' is that window of up to 'n' unacknowledged packets is allowed to be sent into the network. The receiver sends ACK if packet is received in-order and sends NACK or ignores for out-of-order packet. In response to this, what sender does is that if NACK received or in case of timeout for packet n, it begins resending from n all over again. It helps receiver from entering the swamping state.

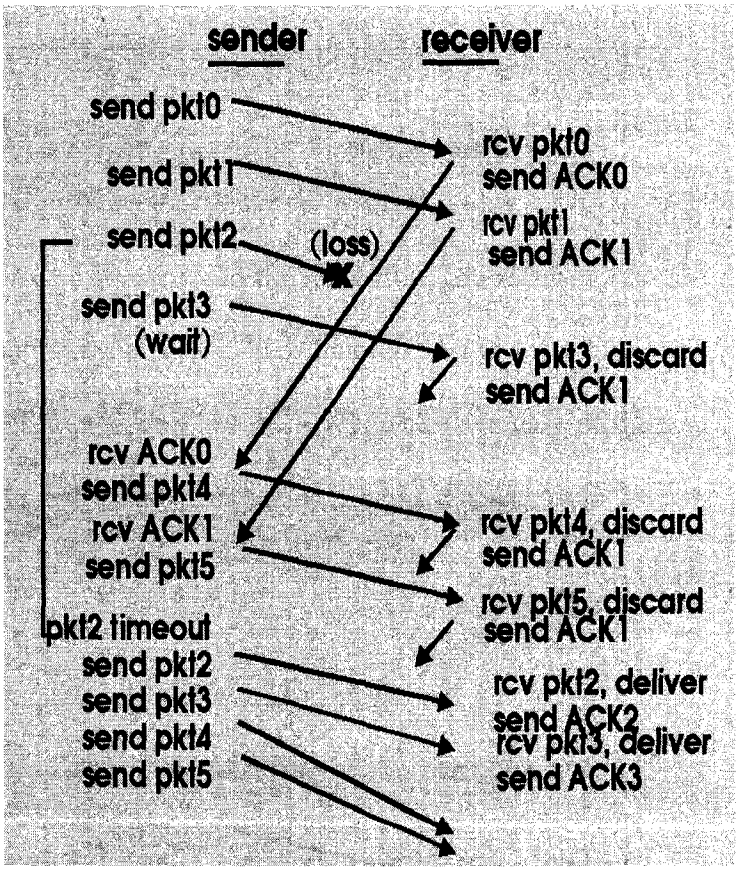


Figure B-2: Go-Back-n example *(Computer Networks - Jim Kurose pp. 63)*

Appendix: C – Network Quantitative - Performance Metrics

Two important quantitative metrics used to evaluate the network performance are, Bandwidth and Latency. Researchers have been highlighting the performance of congestion control algorithm with the metrics throughput and loss rate.

Some of the important network applications are file transfer and remote login. While file transfer is throughput sensitive the remote login application has a greater concerns for delay as well as loss rate.

- **Latency**
  - Propagation Delay – The time required for a packet to travel from source to the destination. Propagation delay is the distance between two end-points divided by the propagation speed.
  - Transmission Delay – Time taken to transmit the packet at the bit rate of link. If the packet is L bits long and the bit rate of the link is R bps. The transmission delay is L/R.
  - Queuing Delay – Also called as buffer delay. This is the time elapsed between en queuing of packet to de-queuing of it.

- End to end Delay – It is defined as the sum of all the delays along the path from source to destination.

Since the RTT consists of propagation delay and buffer delay, different propagation delays influence the frequency of the sending rate.

- **Bandwidth**

Bandwidth is basically a path capacity. It is defined as number of bits that can be transferred over the network per unit of time.

- Bottleneck Bandwidth – In an environment if there is no traffic then bottleneck bandwidth is the bandwidth of lowest-bandwidth link from source to destination.
- Available Bandwidth – The maximum rate that the path can provide to a flow without reducing the rate of the cross traffic.

$$\mathbf{BW = (MSS * C) / (RTT * P^{1/2})}$$

Above equation suggests that Bandwidth is directly proportional to segment size and inversely proportional to delay. Also

$$\mathbf{W = BW * RTT}$$

Hence rate is also directly proportional to these two factors.

- **Throughput**

The amount of data transferred (Number of bits, characters or blocks passing through a network) from one place to another or processed in a specified amount of time.

throughput <= (0.7 * MSS) / (rtt * sqrt(packet-loss)) [Mathis 97]

throughput α MSS (packet size) ∵ MSS (MTU – TCP header)

throughput α 1 / packet-loss α 1/ RTT

- **Loss Rate**

Packet loss is calculated in the number of packets lost/dropped in one RTT. From above equation it is clear that reduction in loss rate will help in improving the throughput.

Appendix: D – Verification of ABCD formula

The topology and traffic configuration of experiments discussed in Chapter 5 does not allow verifying the Three-packet theory. We did some separate experiments with traffic (Constant Bit Rate), which will be independent of other fluctuations. These experiments produce the results which verify the correct available bandwidth.

Experiment 1 - Node 1 is sending UDP packets of 1000 Bytes to the node 4. The application used for generation of traffic is CBR at the speed of 0.5 Mbps. All the links are 10 Mb except the link 2-3 is bottleneck link with the bandwidth 1.5 Mb. Without any other traffic available bandwidth should be 1 Mb (theoretically).

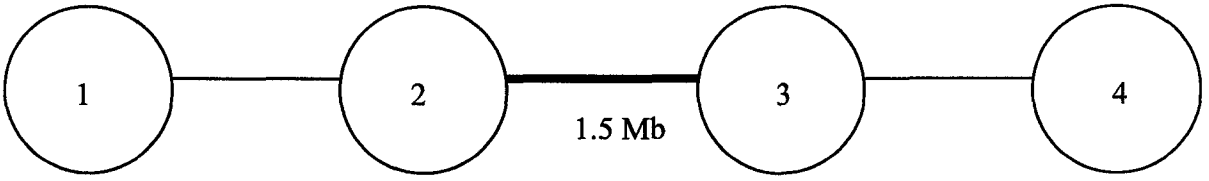


Figure D-1: Topology for ABCD formula verification

We calculated available bandwidth at different instances of 200 seconds simulation and since the traffic is constant bit rate, once the system is tuned we found no deviation in AB from the value 0.9765 Mb i.e. close to 1Mb

Experiment 2- To observe the change in the bandwidth we introduced cross traffic from node 5 to node 6. Nature of the cross traffic is again kept as contact bit rate since with the other sort of cross traffic it's hard to verify the available bandwidth produced by formula. So that it reflects practical conditions.

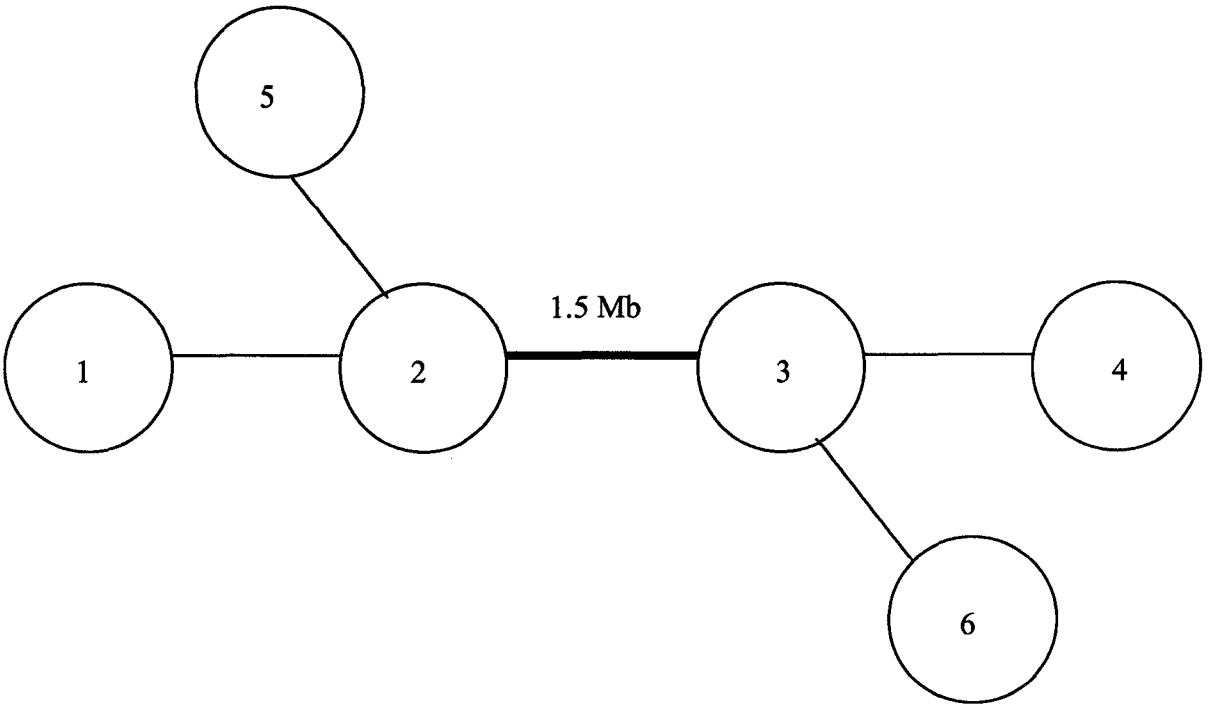


Figure D-2: Topology to observe effect of cross traffic

Node 5 is also sending at the rate of 0.5 Mb/s. We calculated the available bandwidth at various instances of simulation and found that available bandwidth is 0.47 Mb i.e. close to 0.5 Mb.

Vita Auctoris

Aniruddha Bharadwaj was born in Aurangabad, (Maharashtra) India in June 1979. In June 1996 he graduated from Saraswati Bhuwan High School, Aurangabad. In December 2000 he received Bachelor of Computer Science degree from the University of Pune, India. He joined the graduate program at the University of Windsor, Canada in fall 2001. In May 2003 he was elected as a graduate student representative for the University of Windsor SENATE and served on various departmental and university committees. He worked as a Sessional Instructor for the School of Computer Science at the University of Windsor from January 2004 to August 2004. He is scheduled to pursue his doctoral studies at the Wayne State University, Michigan, USA and intends to continue with the career in the field of academia. His research interests include Active Networks, Congestion Control, Grid Computing and Distributed Computing.