"A Packet Drop Guesser Module for Congestion Control Protocols for High speed Networks"

SUBMITTED BY

Ghani-ur-rehman

472-FBAS/MSCS/F08

SUPERVISED BY

Mr. Qaisar Javaid

Mr. Shehzad Ashraf Chaudhry

Department of Computer Science & Software Engineering

Faculty of Basic and Applied Sciences

INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD

(2012)

1 - Internet (computer network)

2 - Intergrated services digital networks

Final Approval

It is certified that we have examined the thesis titled "A Packet Drop Guesser Module for Congestion control Protocols for High Speed Networks" submitted by Ghani-ur-Rehman, Registration No. 472-FBAS/MSCS/F08, and found as per standard. In our judgment, this research project is sufficient to warrant it as acceptance by the International Islamic University, Islamabad for the award of MS Degree in Computer Science.

Dated: 02-May-2012

Committee

External Examiner

Prof.Dr. Abdus Sattar
Former D.G,
Pakistan Computer Bureau,
Islamabad.

Internal Examiner

Prof.Dr. Muhammad Sher
Chairman
Department of Computer Science & Software Engineering
International Islamic University, Islamabad.

Supervisor

Mr. Qaisar Javaid
Assistant Professor
Department of Computer Science & Software Engineering
International Islamic University, Islamabad.

Co- Supervisor

Mr. Shehzad Ashraf Chaudhry
Lecturer
Department of Computer Science & Software Engineering
International Islamic University, Islamabad.

DEDICATION

To my family, my friends and my teachers.

Thanks for understanding the stressful moments I had, for the prayers and support to overcome them and for all the joy you have brought to me.

DECLARATION

I hereby declare that this thesis, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that no portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Ghani-ur-rehman

ACKNOWLEDGMENTS

I would like to thank my supervisor **Mr. Qaisar Javaid** Assistant Professor and Co-Supervisor **Mr. Shehzad Ashraf Chaudhry** Lecturer (Computer Science) for their support and guidance throughout of my study at International Islamic University Islamabad. I would also like to pay special thanks of my friend **Israr Ullah** who help me during my hard times when I need his support during thesis study and simulation.

I would also like to thank my parents, my brothers, who had always given me the courage, best wishes and support during my career.

Ghani-ur-rehman

Abstract

Different high speed Transport layer protocols have been designed and proposed in the literature to improve the performance of standard TCP on high BDP links. They mainly differ in their increase and decrease formulas of their respective congestion control algorithm. Most of these high speed protocols consider every packet drop in the network as an indication of congestion, and they immediately reduce their congestion window size. Such an approach will usually results in under utilization of available bandwidth in case of noisy channel conditions. We take CUBIC as a test case and have compared its performance in case of normal and noisy channel conditions. The throughput of CUBIC was drastically degraded from 50Mbps to 0.5Mbps when we introduced a random packet drops with 0.001 probability. When the probability of the packet drop increases the throughput gets decreased. Indeed, we need to complement existing congestion control algorithms with some intelligent mechanisms that can differentiate whether a certain packet drop is because of congestion or channel error, thus avoiding unnecessary window reduction. In order to distinguish between packets drops, we have proposed an algorithm, a k-NN based module to guess whether the packet drops are due to the congestion or any other reasons. After integrating this module with CUBIC algorithm, we have observed significant performance improvement. Simulation results show that even with 0.001 random packet drop probability, CUBIC protocol was able to achieve throughput about 35 Mbps, using our proposed module, which was otherwise below 0.5 Mbps.

Abbreviations

RAND Research and Development

MIT Massachusetts Institute of Technology

ARPANET Advanced Research Projects Agency Network

DARPA Defense Advanced Research Projects Agency

PSINet Performance Systems International

ISP Internet Service Provider

UUCP Unix-to-Unix Copy

WAIS Wide Area Information Servers

FTP File Transfer Protocol

GPS Global positioning System

PDA Personal Data Assistant

LAN Local Area Network

MAN Metropolitan Area Network

WAN Wide Area Network

TCP Transmission Control Protocol

UDP User Datagram Protocol

SYN Synchronize

FIN Finish

ACK Acknowledgment

MSS Message Segment Size

cwnd congestion window

BDP Bandwidth Delay Product

RTT Round Trip Time

Mbps Mega bits per second

Gbps Giga bits per second

BIC Binary Increase Control

HSTCP High Speed Transmission Control protocol

HTCP Hamilton Transmission Control protocol

CTCP Compound Transmission Control protocol

XCP eXplicit Control Protocol

LTCP Layered Transmission Control protocol

STCP Scalable Transmission Control protocol

AIMD Additive Increase Multiplicative Decrease

RTO Retransmission Time Out

NIC Network Interface Card

SNR Signal to Noise Ratio

VINT Virtual Interchange Test Bed

MAC Media Access Control

k-NN k-th Nearest Neighbor

NAM Network Animator

NS-2 Network Simulator 2

Chapter No	Contents	Page No

List of Tables

List of Figures

Chapter 1

Introduction

INTRODUCTION

1.1 History

That a network is *group of two* or more systems able to *talk* to one another, it can easily be concluded that the history of wired networks predates computers and goes back to telegraph systems and telex machines. Wired networks have evolved through many stages from data communication to digital internet. In the beginning, point to point communication was used between the end systems. The concept of switching was there since the very start of networks, which over the time began to mature. In the beginning circuiting switching was used in all the networks. Networks of that age (1950s and 60s) were only limited to those communication stations which were allowed to communicate. They used gateways and bridges for the switching purpose. That was the time when global communication systems were envisioned for knowledge sharing. Circuit switching, at the time, was found to have many drawbacks and limitations. Thus an active research began for the quest of a superior switching scheme, giving rise to packet switching. In 1960s, Paul Baran of RAND Corporation and Donald Davies of National Physical Laboratory, UK, and independently transmitted information divided in 'message-blocks' across networks. The later named it packet switching. Leonard Kleinrock of MIT developed that mathematical theory upon which this switching worked. Packet switching had many pros as compared to circuit-switched networks like optimal bandwidth utilization and response time. On October 29, 1969 a new milestone in the history of wired networks was laid: the world's first ARPANET link. It was all done by Robert Taylor at DAPRA and Larry Roberts at MIT. The link connected Stanford Research Institute and University of California, Los Angeles. In early 1970s there were so many networking methods and protocols that unification was much needed for the compatibility of these protocols. The idea of TCP/IP was then proposed by DAPRA's Robert E. Kahn. Earlier in 1974 the word 'internet' had come into existence as a shortening of internetworking. In 1989, 'The World', world's first ISP, started working for commercial use in USA. File and information organization tools were the demands of the day during 80s and early 90s. Projects such as gopher, WAIS, and FTP Archive attempted to cater for this. Although they failed, new ways had been for others. As today the internet is browsed and searched, using tools and websites which are very high-end and most optimized. The current trend is to introduce

smaller devices to connect to the Internet. Smaller devices like PDAs, smart phones, pocket PCs and GPS devices are capable to be connected to the internet. Many websites design special pages to work such levels.

With internet enjoying ubiquity, high data-rates, and increased accessibility to all communities, new services like social networking and businesses have grown rapidly which has enabled people from all spheres of life to communicate and share interests with others around the globe.

1.2 Introduction

Network [1] is group of connected nodes that shares resources and information's such as data, hardware and software. A network may be small and large. Small network is one of the simple networks that connects only two computers together through a single cable. A large network may connect hundred and thousands of computers together. Without network, resources can be accessed in own computer such as disks drive, printers and folders etc. These resources are called local resources. It is network that makes it possible to share resources with different of users. All resources can be shared with different computers through network, once a computer is connected to network. The resources may be folder, disk drive, file and printer etc. even internet connection may be shared with other computers. Hub is a device which is mainly used by the most networks to connect computers together. Now a day a network can be made without any hub and cables. In such types of network computers use radio communication for sending and receiving data with one another and is called wireless networks.

1.3 Network classification

Network can be classified according to different characteristics, such as on the basis of coverage area we classify them such as LAN, MAN, and WAN. We can also classify computer network on the basis of topologies such as Bus, star, Ring and Mesh Networks. Network topology is the structure of network including the physical arrangement of the devices.

When classified in terms of connectivity and infrastructure we come up with the picture of Network Structure, as the figure shows below. We can see that Ad hoc network are wireless in which nodes are free to move anywhere in the network and have no fixed centralized

infrastructure. In contrast, Sensor networks are particular type of ad hoc Networks in which communication is directed to a special node called Central or Sink node.

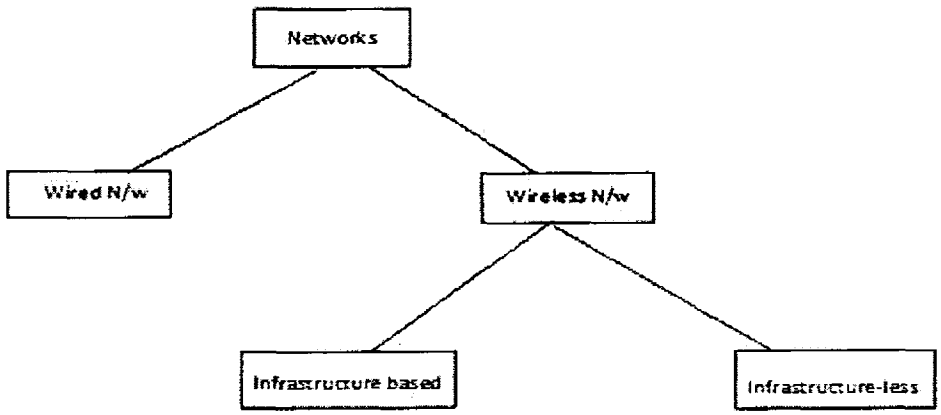

Figure 1.1 Classifications of Networks

1.3.1 Wired Network

Wired networks [2] enable users to move a lot of data so rapidly. Wired networks are more secure, faster, and more affordable than wireless ones. Wired Networks can be made up of three basic components. (1) Computers or similar devices that are connected to one another are called nodes. (2) Network hardware which is used to connects computers to one another, cables and devices which are used to connect all these cables. (3) Network software that runs on each computer and makes it possible to communicate with all other computers in the network. A physical media such as hub, switch and router are also used in wired networks to connect two or more computers which increase the strength of the connection.

1.3.2 Wireless Network

Wireless Network [2] refers to the kind of network that connects different computers without having any types of cables between them. Wireless network transmit data using some types of radio frequencies in air. These networks are more flexible and can be configured easily. Wireless network can be setup easily and fast. They can be infrastructure based and infrastructure-less. Infrastructure based require a fixed centralized infrastructure such as router

and access point between nodes while infrastructure-less do not need any centralized infrastructure and may be deployed with ease.

1.3.3 Ad hoc Network

In Ad hoc Networks [2] all the nodes are peers, which means that each node participates in routing by forwarding data to other nodes in the network. Every node in this network serves as source and destination as well as the router for packets delivery. These Networks do not need any fixed centralized infrastructure so they can be deployed easily with low cost. Many challenges may come in the way while designing these networks despite low cost and easily deployed such as energy management, security, self-organization, and scalability etc. The followings are the some applications of Ad hoc networks.

- Military applications
- Collaborative and Distributed Computing
- Emergency operations

1.3.4 Sensor Networks

A Sensor Networks [2] is a network of sensor devices which usually are small in size and inexpensive in price to monitor and collect physical or environmental data. Due to the Ad hoc method of deployment and inexpensive nature, there are certain power and computational limitations in sensor networks. The followings are the common applications of sensor networks.

- Environmental monitoring
- Habitat monitoring
- Seismic detection
- Military surveillance

1.4 Transport layer

The main duty of the transport layer is to guarantee reliable delivery of the message without any error. The functions performed at the transport layer are the reliability, flow control, error control and congestion control etc. When sending rate by the sender is greater than the receiver then it is called flow control. Error control means to deliver data without any part of it is

damaged, duplicated and out of order. When number of packets sent to the network exceeds the network capacity then it is said to be in state of congestion. Transport layer also uses acknowledgment system to ensure reliability.

1.5 Transport layer protocols

The most famous protocols of Transport layer are Transmission Control Protocol and User Datagram Protocol.

1.5.1 Transmission Control Protocol (TCP)

TCP [3], [4] is one of the reliable and connection-oriented protocols. A connection is established between a sender and receiver to transfer data. The data may be sent in both directions. When data is delivered successfully a connection is then terminated. The connection creation and terminated is done by using Three-Way Handshaking process. The Three-Way Handshaking process for connection creation is explained below as follows.

- A client sends a TCP SYN message to the server. The SNY message does not have any data but it is used only for sequence number. The server receives the SYN message of the client.
- Server then sends a SYN plus an acknowledgement message to the client. The SYN message for communication and an acknowledgement message are for the receipt of client SYN message.
- When client receives the SYN message of the server, then it sends an acknowledgment message to the server and hence the connection is created.

The connection is terminated using Three-Way handshaking process. The followings steps may be used for connection termination.

- A sender sends a FIN message to the server. The FIN message is used for sequence number if it has no data. Sometime it contains the last chunks of data.
- When the FIN message is received by server from the client, it sends FIN plus an acknowledgment message to the client. The FIN message is used for termination of the

communication and an acknowledgment message is used to inform client about the receipt of FIN.

- At last, after receiving the FIN message of the server, the client sends an acknowledgment message to the server and hence the connection is closed.

TCP also provides reliability. In TCP sender sends data to the receiver. After receiving data from the sender, receiver informs it by sending an acknowledgment that the data is received without any error. Transmission Control Protocol has three important aspect namely congestion control, error control and flow control. Different types of window mechanisms are used for handling flow control by TCP. It uses various kinds of error control mechanisms to detect errors such as loss of packet, packet that is corrupt, duplicate packet and packet that is not in proper order. The congestion control aspect of the TCP is explained in detail in the next section.

1.5.1.1 Congestion Control in TCP

Network congestion [5] refers to the situation in which the capacity of a network is exceeded by the number of packets sent to it. It may mean load on the network. When this load is kept below the total capacity of the network, it is called congestion control. Congestion is possible in any system that involves waiting. In network congestion occurs because routers and switches have queues which store the packets. We can easily understand congestion control by taking an example of congestion control in TCP.

The TCP is an acknowledgement based protocol in which a receiver must sends an acknowledgment to the sender after receiving a packet. Sender can only send a new packet after an ACK has been received from receiver. One of the important duties of the TCP is congestion control. The TCP handle congestion in three steps: slow start, congestion avoidance and congestion detection. All these are discussed in the next section.

This slow start has an exponential increase. The increase is dictated by the size of the congestion window (maximum number of packets that can be transmitted at a particular time) which starts with one maximum segment size (maximum amount of data that a segment can hold). During the connection creation the maximum segment size is determined by using the option of the same name. When an acknowledgment is received to send segment the size of the

cwnd (maximum number of packets that can be transmitted at a particular time) is increased to maximum segment size. As the name implies the window starts slowly and increases exponentially. For example the sender starts with congestion window is set equal to one maximum segment size means that sender can sends only one segment. For example we have seven segments, When an ACK is received for segment I, the size of the window is incremented by 1, which means that the value of cwnd is now 2 and 2 more segments can be sent. When an ACK is received for those two segments, the size of the window is incremented by one MSS for each segment. When all seven segments are ACK the congestion window is equal to 8. When there is delayed acknowledgement, the size of the window is incremented less than power of 2. Slow start cannot continue infinitely and stops when it reaches the threshold.

The size of the window in slow start phase is increased exponentially. To avoid congestion, TCP defines congestion avoidance algorithm, which uses additive increase. When cwnd size reaches threshold, slow start stops and additive increase start. The additive increase algorithm, the size of cwnd is incremented by 1 additively until congestion is detected.

If congestion occurs, the size of the cwnd must be decreased. When sender knows that congestion has occurred, it retransmits a segment. The segment can be retransmitted for two reasons. The first one is when the timer time is out and the second is by receiving three duplicate ACKs. The threshold's size is halved due to these reasons. For the two reasons the size of the threshold is reduced to one half (multiplicative decrease). Thus, there are two choices for the TCP:

1. If a time-out happens, the possibility of congestion is very high. Here TCP has a strong reaction.

- Set the value of threshold to $\frac{cwnd}{2}$.
- Set cwnd to the size of one segment.
- Start with slow-start step again.

2. If three duplicate ACKs are received, the chances that congestion occurs are less. A network segment may have been dropped but not all the segments, because three duplicate ACKs are received for the segments after this one show that they are received safely. This is called fast

transmission and fast recovery. For fast retransmission and fast recovery the size of the congestion window must be four packets or more. In this case TCP has not a stronger reaction:

- Set the value of threshold to $\frac{cwnd}{2}$.
- Set cwnd to the value of threshold.
- Start the congestion avoidance phase.

Congestion detection may occur in one of the followings two ways.

- If detection is due to first case, a new slow start phase begins.
- If detection is due to second case, a new congestion avoidance phase begins.

The increase and decrease formula for TCP is given below.

1. Increase when all acknowledgment have been received for the window, which means fully acknowledged:

$$cwnd = cwnd + 1 \quad \text{.....(1.1)}$$

2. Decrease when acknowledgment for message is not received which means the message is lost:

$$cwnd = cwnd - \frac{cwnd}{2} \quad \text{.....(1.2)}$$

1.5.2 User Datagram Protocol (UDP)

UDP is one of unreliable and connectionless transport layer protocols. The user datagram does not depend on some other datagram sent by UDP. There is no contact between user datagram's even if they are delivered by the same sender and received to same receiver. So it does not have any connection establishment and connection termination process. Like TCP user datagrams have no numbering facility and may pass through various routes. There is no error control and flow control techniques in UDP. Senders sends messages to the receivers and does not know that the messages are damaged, corrupted, received out of order, or lost. It uses

checksum for error detection. When sometime errors are detected, it simply discards the message.

1.6 Problems with TCP

One of the most common problems with TCP [5] is that it is not best suited on high bandwidth-delay link. Because the congestion control algorithm of TCP takes much more time to recover from error and to efficiently use the available bandwidth in high-BDP network. The second main problem of TCP is that its performance is not found to be satisfactory when there is a concurrent bursting flow and some random loss. Such types of problems do not allow TCP to best utilize the bandwidth. For concurrent flows which have different RTT, TCP also face the fairness problem.

1.7 High Speed Networks and Protocols

High speed networks refer to the networks that usually have higher rate of data transmission such as high speed LAN and Ethernet. The rate may be varying from few Mega bits per seconds (Mbps) to Giga bits per seconds (Gbps). The common applications of high speed networks are telemedicine, videoconferencing and weather simulations etc. High speed network may perform better in the situation where the end system regulates their flow of data for using the available network resources efficiently without more loads on the system. When there are more loads on the systems, it leads to congestion and throughput collapse. Simply, the high speed networks are introduced to sends a large amount of data quickly. As TCP is not best suitable for high-speed networks, for this purpose different high speed network protocols have come into existences which are called the variants of the TCP. All these high speed network protocols are discussed in the next chapter of the literature survey.

1.8 Major Research areas of HSN

Due to its high speed nature, High speed networks also face some major challenges. The first one is the networks communication measurements. The second major research area is the understanding different types of attacks. Network security is also said to be the major challenge in the high speed networks. Most of these challenges are addressed to some extent. Research is underway on some of them.

1.9 Objectives

Most high speed TCP variants treat every packet drop as an indication of network congestion and use the decrease formula in reaction. This results in their degraded performance, especially in noisy channel conditions. The objective of this thesis is mainly to develop some intelligent mechanism that can enable these protocols, CUBIC in this case, to differentiate between packet drops due to congestion and noise/error in the channel. The idea is to make use of history and identify patterns to correctly guess whether the packet loss is due to error in the channel, transmission impairment or congestion in the network. One possible solution is to estimate whether the dropped packet was due to congestion or any other reason is to decide whether or not to reduce the congestion window, or *cwnd*. We use the k-nearest neighbor algorithm (k-NN) for such packet drop estimation. The objectives of this work are as follows:

- Carefully design a history based supporting module on the basis of pattern matching in packet drops trend, to guess about the reason of packet loss.
- Incorporate this module into the CUBIC protocol to verify the algorithm and see its effect on the performance of the protocol.

All simulations will be performed using Network Simulator or NS-2.

1.10 Organization of Thesis

In Chapter 2 a detailed literature review is performed to highlight different high speed protocols, their advantages and disadvantages.

In Chapter 3 we present the problem faced in high speed protocols and take CUBIC as a test case to check its performance in normal network condition as well as noisy channel conditions in detail.

Chapter 4 contains proposed solution, proposed solution model and proposed solution algorithm.

Chapter 5 contains different simulation results. All the simulation will be performed using famous Network Simulator NS-2.

Chapter 6 is composed of the conclusion and future work.

Chapter 2

Literature Survey

LITERATURE SURVEY

2.1 Introduction

Conventional TCP is not a good candidate for high bandwidth delay link. TCP's congestion control algorithm takes long time to take benefit of large bandwidth and to recover from the loss. The TCP has the followings main drawbacks. The first main problem with the TCP is that it considers the entire packet drop due to the network congestion. When a packet is lost in the network it cuts the size of the window into half. If there is further packet drop in the network it again cuts it into half. All packets drop may not be due to network congestion only, but may be due to some type of noisy errors. The second drawback of TCP is that when the RTT from source to the destination and back increases it operates progressively slower. A number of high speed protocols are proposed to improve the traditional TCP performance on high speed network [6].

2.2 Types of High Speed Protocols

The high speed protocols can be further classified into two types, loss-based and delay based. Protocols that depend on packet drops due to congestion detection are referred to as Loss-based, and the delay based are those protocols that depend on queuing delay to collect information about congestion detection. FAST is the only delay based protocol. Loss-based are further classified into two types: first, those that take the previous congestion window into consideration for calculating the next congestion widow, second, those that are time-based, since last packet dropped for calculating the next congestion window.

The followings high speed networks protocols have been selected for the surveys which are discussed in detail below.

- Binary Increase Control TCP
- Cubic TCP
- Fast AQM Scalable TCP
- High Speed TCP
- Compound TCP

- Scalable TCP
- TCP Vegas
- Hamilton TCP
- Layered TCP
- Explicit Control Protocol

1. Binary Increase Congestion Control shortened as BIC was developed by Injong Rhee, and Lisong Xu [8]. It was developed at North State Carolina University. The driving force behind developing BIC TCP was to improve the TCP's effectiveness on links with high bandwidth delay product. BIC TCP uses two schemes of window size control, namely additive increase and binary search increase.

In Binary search increase scheme the congestion control is considered to be a searching problem. It provides 'yes' or 'no' feedback for the packet loss. There are two starting points for this search. The first one is the current minimum window size Wmin (size of the window just after the reduction). At this point the cwnd is free of any losses. The second one is maximum window size Wmax (the size of window just before the reduction) where the available link bandwidth is utilized by the cwnd. Binary search increase repeatedly calculates the midpoint between Wmax and Wmin called target window size. This midpoint is the set as the current size. It also keeps watch on feedback as packet loss. The feedback being 'yes', means that the packet is lost. Then the midpoint is set to the new Wmax. But when feedback is 'no' then it means that there is no packet loss. In this case the midpoint is set to the new Wmin. This process continues until the difference between the size of the window just after the reduction and the size of the window just before the reduction is below the preset threshold value called the minimum increment (Smin). This process is called binary search increase. Binary search increase is more aggressive at the start, at a time when the difference between the current window size and target window size is very large. Similarly, for this difference being small, the binary search increase is less aggressive. One of the important properties of the protocol is that its increase function is logarithmic. It can also reduce the chances of the packet loss. The main advantage of the binary search increase is that it provides a concave response function [7], [8].

Mixed with additive increase, the binary search increase gets faster convergence and RTT-fairness. With a large distance from the current minimum to the midpoint, the window size may be increased to that midpoint, resulting in more stress that the network observes. When the current window's size distance to a target in the binary search increase is larger than the S_{max}, the window's size is increased by S_{max}, being maximum increment. This is done until the distance becomes less than S_{max} at time the window increases to the target after a large amount of window reduction. In the beginning this policy increases the window linearly and then logarithmically. The combination of binary search increase and additive increase is called binary increase. When the window is large and multiplicative decrease policy is mixed with the binary increase becomes an additive increase. When the size of the window is larger, there is a large reduction in the multiplicative decrease and hence long additive increase period. But for the size of the window being small then it is approximately equal to the binary search increase and hence shorter additive increase period.

In Multiplicative decrease when packet lost happens, there will be a reduction in the size of the cwnd by a multiplicative factor of β, which is commonly 0.875 used by others protocols.

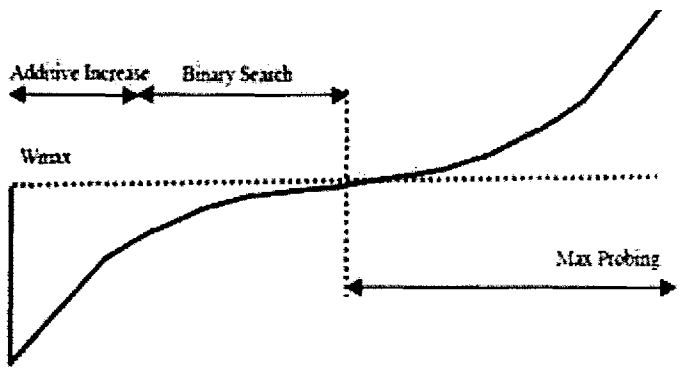

Figure 2.1 Window Growth function of BIC TCP [8]

Of the many advantages of the BIC TCP, the main advantage is that due to the use of the binary increase policy it uses the bandwidth in the most optimum way. The main disadvantage of the BIC TCP [8] is that there are early packet drops on its initial ascent because it makes wrong

estimation of maximum link capacity. It may not fairly share its link with other because of the fast convergence strategy. Both these schemes are shown in the figure above.

2. CUBIC TCP was proposed by Shashank Jain and Gaurav Raina [9], [10]. Considered to be a very TCP friendly variant of the original TCP, CUBIC can be regarded as the extension of BIC protocol as it the BIC variant with many new features that add to the usefulness of BIC. The window control of the BIC TCP can further simplified by the CUBIC TCP. Although BIC guarantees good stability, fairness and scalability during current high speed TCP variants, its growth function is not that good for TCP under smaller RTT and low speed networks. The window control different phases make it too complex to understand the protocol easily.

To discuss CUBIC protocol [13] in detail the followings terms are used.

cwnd: value of the congestion window

C: scaling constant

T: time to denote the last congestion event

Wmax: it denotes the size of the cwnd

β: multiplicative decrease factor, which is normally 0.8

$$K = \left(\text{Wmax} \frac{\beta}{C} \right)^{\frac{1}{3}} \quad \text{.....(2.1)}$$

When an acknowledgment arrives:

$$cwnd = C(T-K)^3 + W_{\max} \quad \text{.....(2.2)}$$

Packet loss means no acknowledgement is received:

$$cwnd = \beta \text{ Wmax} \quad \text{.....(2.3)}$$

The window growth function of CUBIC protocol is cubic function and its shape is very much the same as the growth function of the BIC as shown in the figure below. The CUBIC growth function grows very fast at Wmax with the origin. It may grow slower when it nears the Wmax. Around the Wmax the window increment becomes 0. Above Wmax, CUBIC starts checking more bandwidth available in which the window gradually increases at the start and makes its growth faster when it moves away from Wmax. This slow growth ensures modification in the stability of the protocol while the fast growth ensures the scalability of the protocol. It also ensures the intra protocol fairness among different flows of the same protocol due to the cubic

function of CUBIC protocol. As the rate of growth is dominated by the elapsed time t, the cubic function also provides good RTT fairness property [12]. This ensures linear RTT fairness of different competing flows. Limit is imposed on window increment so that it is less than or equal to S_{max} per second to add something more to the fairness and stability property. Due to this feature the window will grow linearly when it is not near to the Wmax. Under small RTT, the CUBIC is fairer than other high speed networks protocols.

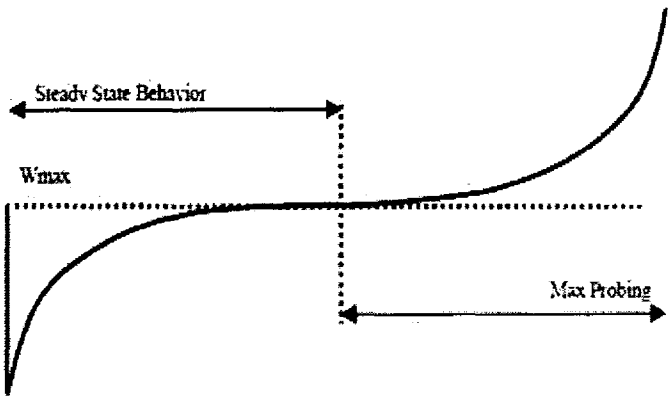

Figure 2.2 Window growth function of CUBIC [8]

Of the many other advantages of CUBIC [8], it main advantage is that it modifies the fairness characteristic of BIC while keeping its scalability and stability intact. The main disadvantage of the CUBIC [13] is that it is so slow in increasing its congestion window to completely occupy the link.

3. FAST was proposed by C. Jin, D. X. Wei, and S. H. Low, Hegde [14] is a delay-based protocol. FAST continuously monitors the flow's RTT so that it is able to check the congestion level in the network. The congestion window is updated by FAST and every other RTT based on the RTT observed and α which contains the number of packets it tries to maintain in the router queues. FAST is not a good performer because the size of the router queue being less than α. When the observed minimum value is not changed it aggressively increases the congestion window. In FAST the congestion control of TCP is composed of four important components. These are shown in the figure below.

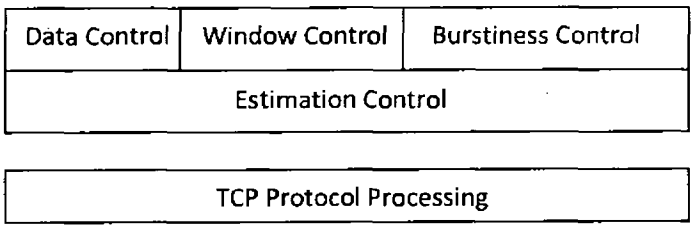

Figure 2.3 FAST TCP Architecture [6]

These four components are not dependent upon each other. All the components are designed separately. The work of data control component is to examine which packet to transmit, the window control examines how many packets to transmit, and the burstiness control examines when to transmit these packets. The estimation component provides some meaningful information. The other three components can make decisions upon this provided information. The estimation component also gives two types of information for the packet being sent. The data control component chooses the next packet to send from three available options: new packet, negatively acknowledgement packet and packet for which an acknowledgement is not yet been received. Window control sends packets in order at RTT timescale.

In FAST, the network consists of a set of resources with limited capacity. These resources are transmission link, processing unit and memory etc. Different unicast flow shares the network. These unicast flows are identified by their sources. The main problem faced in this protocol is the unfairness and the instability in small buffer or in long delay networks [13], [15], [26].

Now the algorithm of FAST discussed in detail below [13].

AvgRTT: current average RTT

baseRTT: the minimum RTT

γ: ranges from 0+ to 1.

cwnd: value of congestion window

qdelay: weighted queuing delay

The followings equations can be used by window control component to manage *cwnd*:

$$cwnd = \min\left\{ 2cwnd,\ (1-\gamma)cwnd + \gamma\left(cwnd\left(\frac{\text{baseRTT}}{\text{avgRTT}}\right) + \alpha\left(cwnd, qdelay\right) \right)\right\} \quad \text{.....(2.4)}$$

$$\alpha\left(cwnd,\ \text{qdelay}\right) = \begin{cases} \alpha \times cwnd & \text{, when } qdelay = 0 \\ \alpha & \text{, when } qdelay \neq 0 \end{cases}$$

4. HS-TCP was proposed by Sally Floyd [16] especially for large congestion window. It also solves the standard TCP's shortcoming of getting a large congestion window in a situation where the packet loss rate in not too much high. HS-TCP also proposed a little change in the increment and decrement parameters of the standard TCP. It also makes an attempt to improve the loss recovery time of the traditional TCP by bringing some modifications in its AIMD algorithm. This enhanced algorithm affects the higher congestion window. For the congestion window being lower than the value of the threshold, called low window, then the standard AIMD algorithm is applied. Otherwise it applies high speed AIMD algorithm. The important and demanding task of HS-TCP is to makes its flows more aggressively having no response from receiver or link router. HS-TCP is specifically designed for low loss rate in a high bandwidth environment and tries its best to be more aggressive than standard TCP. Its low loss rate is 10-2. Traditional TCP has a response function of $\frac{1.2}{\sqrt{p}}$ where p denotes the loss rate. HS-TCP [17] has a response function of $\frac{0.12}{(p^{0.835})}$. It can be concluded that the response function of the HS-TCP is more aggressive than that of the standards TCP's response function for the packet rate being less than 10-3. The default values of a and b are .01 and .875 respectively.

The congestion control algorithm of the HS-TCP [13] contains the followings terms.

cwnd: current congestion window

Low window: congestion window threshold whose value is larger than cwnd

a (cwnd): Increase parameter

b (cwnd): Decrease parameter

When an Acknowledge arrives;

cwnd = cwnd +1 cwnd<= low window and slow start

$= cwnd + \frac{1}{cwnd}$ cwnd<= low window and congestion avoidance

$= \text{cwnd} + \frac{a(cwnd)}{cwnd}$.....(2.5) cwnd> low window

When a packet is lost;

cwnd = 0.5 cwnd cwnd<=low window

$= \left(\left(1-b\ (\text{cwnd})\right)\ \text{cwnd}\right)$.....(2.6) cwnd> low window

The above algorithm is known as additive increase multiplicative decrease (AIMD) having increase and decrease parameters. The values of increase and decrease parameters are determined by the lookup table. The main advantage of the HS-TCP [18] is that it is more aggressive when the loss rate is low and the utilization of the link is very good in the single flow. The main shortcoming of the HS-TCP is that it is not so much friendly with the common TCP flows and converges very slowly.

5. Compound TCP was proposed by Xiuchao Wu [19] to simultaneously fulfill TCP friendliness and efficiency requirements. It is both loss and delay based protocol. The governing idea behind compound TCP is that in case of inefficient utilization of the link, the high speed protocol becomes aggressive in sending the data. But if the link is already efficiently utilized then this situation is no longer valid and being aggressive may cause problems such TCP bias.

It can be noted that the approaches based on delay already have this good property to adjust the aggressiveness based on the utilization of link. However, being inferior to the approaches base of packet loss is the major drawback of the delay-based approaches.

Compound TCP is scalable as well as TCP-friendly:

1. CTCP utilizes the network resources efficiently and achieves high link utilization. Theoretically, CTCP is quick in getting free network bandwidth, using rapid increase rule in the delay-based component, e.g. multiplicative increase.
2. CTCP has fairness of TCP. CTCP limits the sending rate when the link is fully utilized, using the delay based component.

It should be noted that CTCP has much improved RTT fairness in comparison to the regular TCP. This is due to the delay-based component as employed in the congestion avoidance algorithm of CTCP.

The state variables used to implements the compound TCP are congestion window (cwnd), delay window (dwnd), receiver advertisement window (awnd). The sending window of the TCP is calculated below.

$$\text{Win} = \min\left(cwnd + dwnd, \text{awnd}\right) \quad \text{.....(2.7)}$$

The cwnd is changed in the similar fashion as in traditional TCP. In the situation in which an acknowledgement is received, the cwnd is changed as:

$$\text{cwnd} = \text{cwnd} + \frac{1}{(\text{cwnd} + \text{dwnd})} \quad \text{.....(2.8)}$$

6. Scalable TCP shortened as STCP was proposed by Tom Kelly [20]. It is the enhanced version of High-Speed TCP or HS-TCP. The scalable TCP's basic goal is the improvement of loss recovery time, not catered for in the standard TCP. As mentioned earlier the scalable TCP being the enhanced version of HS-TCP, takes its main idea from HS-TCP. To compare and contrast the scalable TCP and standard TCP and HS-TCP: In standard TCP and HS-TCP connections, the time for packet loss recovery depends directly on the RTT and the size of the connection window. On the other hand, in scalable TCP, the packet loss recovery time depends only upon the RTT and *not* on the size of the connection window. With all the other changes and modifications, the standard TCP's slow start phase is not modified for the scalable TCP [21]. The scalable TCP increases the size of its congestion more speedily than the standard TCP while decreases it less speedily than the standard TCP. As in HS-TCP, the STCP has set a threshold for the window size and whenever the size of the cwnd is more than this threshold window size, STCP is used otherwise the standard TCP is used. The threshold window's is set to 16 segments, as the default value. STCP is deployed incrementally. Its behavior is exactly the same as the

parent SCPT for congestion window size lower than the threshold. It has the ability to double its sending rate for any other rate in 70 RTTs and thus the name *scalable TCP.*

The literature of Congestion Control Algorithm [13] of the Scalable TCP mainly uses the terms that follow below:

cwnd : It stands for Current congestion window

LowWindow: Congestion window's size being less than the threshold thus leading to STCP's behavior similar to TCP.

Congestion control algorithm of STCP is not much more complex. Congestion window is increased by 0.01 for every acknowledgment. When a congestion event is detected then congestion window is decreased by 0.125. Packet loss identifies the congestion events.

Upon the reception of ACK:

$cwnd = cwnd + 1$ $cwnd <= LowWindow$ and slow start

$= \text{cwnd} + \frac{1}{cwnd}$ $cwnd <= LowWindow$ and congestion avoidance

$= cwnd + 0.01$.....(2.9) $cwnd > LowWindow$

When a packet loss occurred;

$cwnd = 0.5cwnd$ $cwnd <= LowWindow$

$= (1 - 0.125)cwnd$.....(2.10) $cwnd > LowWindow$

STCP [13] performs better even if the buffer size is small. Upon packet loss, STCP increases its congestion window size speedily and attains the maximum bandwidth available in a very short time. The Scalable TCP [20] at times is not able to maintain the fairness equilibrium, which is one of its drawbacks. Also it shows RTT biasness and has inclination towards TCP.

7. TCP Vegas was proposed by Habibullah Jamal and Kiran Sultan that uses packet delay instead of packet loss to determine the size of congestion window [22]. Unlike other congestion control algorithms which take proper measures only after a packet drop has occurred, the TCP Vegas is sensitive to increase in the RTT. The TCP Vegas extends its retransmission mechanism in the following ways: with the transmission of each segment, system clock is read and then recorded; at the arrival of ACK, clock is again read. RTT is calculated using this time. The timestamp is recorded for the relevant segment. This is a more accurate RTT, using which the retransmission can be calculated as:

a. When a duplicate acknowledgement is received, the algorithm confirms whether the new RTT is greater than RTO or not, where RTT is current time minus timestamp recorded. If greater, the Vegas retransmit the segment without waiting for the third duplicate acknowledgement.

b. On the reception of a non-duplicate acknowledgement, whether the first or second one after a retransmission, Vegas checks again to see if RTT is greater than RTO; if it is, then the segment is retransmitted. Thus, there are some ACKs, that help Vegas determine whether the timeout should happen or not.

Vegas is implemented exploiting the idea of controlling and measuring the extra data that the connection has in transit. The term extra amount of data refers to the data that would have been impossible to send had the bandwidth used by the connection been exactly matched by the available bandwidth of the link. The goal of Vegas is the determination of the proper amount of extra data supposed to be there in the network. So, if a connection sends excessively extra data, congestion will result; on the other hand, if it sends very less extra data, rapid response to the transient increase in the available bandwidth becomes very hard. Research has shown that the efficiency of TCP Vegas is much higher than that of its counterpart TCP Reno causing much less packet retransmissions.

The RTT of the first segment sent is called the base RTT. When the traffic does not overflow the connection then expected throughput is equal to the window size (size of current cwnd) of the base RTT.

$$\text{Expected Throughput} = \frac{\text{Window Size}}{\text{Base RTT}} \quad \text{.....(2.11)}$$

The Actual Throughput per RTT is given by,

$$\frac{W}{RTT} \quad \text{.....(2.12)}$$

The cwnd is adjusted when there is a difference in the actual and expected throughput,

$$Difference = (\text{expected} - \text{actual}) baseRTT \text{.....(2.13)}$$

Also two types of threshold values a and b are defined i.e. a < b and a > b, a < b which means there is too little extra traffic found in the network. The > b means there is too much extra terrific found in the network. When the difference is <a, the cwnd in the coming next RTT is incremented linearly by Vegas. When the difference is < b, the cwnd in the coming next RTT is incremented linearly by the Vegas. When the difference is < a, < b it means there is no change in the cwnd.

8. H-TCP was developed by Pankaj Sharma [13], is an extension of the standard TCP protocol. The H-TCP is better than its parent TCP by having better mathematical and empirical foundations. The proposed improvements in the H-TCP are assumed to behave regularly, like the TCP for LAN and WAN networks with RTT and link capacity restricted to some specific limit. However, the H-TCP is much more aggressive and flexible for fast networks over long distances. It shows flexibility in terms of the adjustment for the available bandwidth and avoidance of packet loss.

The congestion window is incremented by adding some constant value α and decreased linearly by β. By default α=1, and β=0.5. In the case of H-TCP [13], calculating the congestion window is not the same easy as α and is not β constant here but dynamically depending on the differences of time between the different congestion events rather than on previous congestion window values.

In this discussion [13] the followings terms should be used:

α: Increase Parameter

β: Decrease Parameter

Δ: Time since last congestion event

RTTmin: Minimum Round Trip Time

RTTmax: Maximum Round Trip Time

cwnd: Congestion window value

On the reception of an ACK:

$$(i)\ \alpha = 1 \quad \Delta \leq \Delta L$$

$$= 1 + 10(\Delta - \Delta L) + \left(\frac{(\Delta - \Delta L)}{2}\right) 2 \quad \Delta > \Delta L$$

$$(ii)\ \alpha = 2(1-\beta)$$

$$(iii)\ cwnd = cwnd + \frac{\alpha}{cwnd} \quad \text{.....(2.14)}$$

When packet is lost:

$$(i)\ \beta = \frac{RTT\min}{RTT\max} \beta \in [0.5, 0.8]$$

$$(ii)\ cwnd = \beta * cwnd \quad \text{.....(2.15)}$$

The algorithm is said to be an AIMD algorithm.

9. Layered TCP shortened as LTCP was developed by Sumitha Bhandarkar, Saurabh Jain and A. L. Narasimha Reddy [23] is one of the sender-side adjustments to the response function of the standard TCP and tries its best to makes it more scalable in high speed network. The response function of the cwnd of the Layered TCP is clearly defined in two extents, at macroscopic level and at microscopic level. At macroscopic level, Layered TCP employs an idea of the layering to speedily and efficiently explore for the available bandwidth. At the microscopic level, Layered TCP extends the present AIMD algorithm of TCP to find out the per acknowledgment performance. The primary purpose of the LTCP is to make the size of the congestion response function in high speed network under the followings conditions:

a) All LTCP flows which have same RTT should be fair to one another.

b) For the window size not being more than threshold, Layered TCP flows should be fair to the standard TCP flows.

This threshold defines some rule under which Layered TCP is found to be so friendly to the implementation of the standard TCP. In high speed network the selection of window scale makes able the receiver to announce the size of window too much large. In order to make certain that Layered TCP is fair to standard TCP, all the new connection of

the Layered TCP that starts with only single layer and performs the same as standard TCP. The response function of the congestion window is changed only when this window is increased ahead of the threshold.

When congestion window of the Layered TCP flow grows ahead of the LTCP window threshold, the increase performance is changed to perform two dimensionally [23].

a) The layers are increased in the case when the congestion is not practical for more time.

b) The per-ACK congestion window performance of TCP is comprehensive so that a flow can increase its congestion window when working at higher layer faster than compared to working at lower layer. The Layered TCP also called an ACK-clocked and with incoming acknowledgment LTCP flow changed its congestion window. Though LTCP flow increasing the congestion window forcefully compared to the implementation of common TCP based on the layer at which it works. The layers are added if no congestion is observed over an extended period of time. To do this, a simple method of layering is used. When the current congestion window gets higher than the window corresponding to the last addition of a layer, a new layer is added. The main advantage is that "At macroscopic level, Layered TCP employs an idea of the layering to speedily and efficiently explore for the available bandwidth". At the microscopic level, Layered TCP extends the present AIMD algorithm of TCP to find out the per acknowledgment performance.

10. XCP was developed by D. M. Lopez-Pacheco and C. Pham [24], stands for Explicit Control Protocol, is the feedback-based congestion control scheme. For congestion being there in the network, it applies direct and precise router feedback to avoid it. It is especially developed for the purpose of scalability and generality. Explicit Congestion Notification router is used in this protocol to immediately inform sender about the congestion in the network. It shows better performance in high delay bandwidth product network. XCP uses router-assistance to exactly notify the sender about the congestion.

The resource allocation function is divided between a fairness controller and congestion controller by XPC. The duty of the congestion controller is to make sure that flows make use of all the accessible capacity on other hand the duty of the fairness

controller is to fairly allocate the capacity to all the flows. The majority of congestion control schemes are not able to perform this division. In XCP, the implementation and explanation of these two resource allocation functions is made possible due to this division. All the other information about the current RTT and throughput assigned to the router in each packet are sent by the XCP sources. Feedbacks are put into the packet by the XCP routes.

In XCP, every data packet has a congestion header which consists of the size of the current cwnd of the sender, the estimated RTT and a feedback field. The current congestion window size and feedback field of the sender is denoted by H_cwnd field and H_feedback respectively. The H_feedback can be changed at every hop. It depends on the value of the two preceding field. The H_feedback field contains positive or negative values. These values show the quantity by which the size of the cwnd of the sender can be incremented or decremented. With data packets being received, the receiver copies the congestion header and sends an acknowledgment packet to the sender. All these acknowledgment packets do not follow the same routes. On the reception of acknowledgment packets, the source renews its size of the congestion window by the following formula:

$$cwnd = \max\left(\text{cwnd} + \text{H\_feedback};\ \text{packetsize}\right) \quad \text{.....(2.16)}$$

The EC's (Efficiency Controller) main duty is to make the best use of link utilization. It also minimizes the rate of the packet drop. The basic advantage of the XCP is that it has good convergence time to full utilization and fairness to other flows. The main drawback is the high dependence of XCP [24] on the returned ACK packets for maintaining a logical view of the network conditions. This affects the performance of the XCP. It does not make XCP stable, when an acknowledgment is lost on the reverse route in a very high speed networks. XCP tries to keep RTT per connection, requires router participation, deployment, not fair in the direction of the connections with longer RTT. Any malicious sender can falsely compute the header and feedback may lead to errors.

2.3 Summary

In this chapter different types of High speed protocols were discussed. All these protocols consider every packet drop as an indication of network congestion. Packets may

not drop only due to the network congestion; it may be due to some other reasons which will be discussed in the next chapter. In the next chapter of the problem definition we take CUBIC as a test case and check its performance in noisy channel condition.

Chapter 3

Problem Definition

PROBLEM DEFINITION

3.1 Introduction

Packet drops are mostly considered as an indication of network congestion. This assumption is followed by many protocols. Certainly, there are other possible reasons of packet drop which are ignored by these protocols. This often results in poor performance and demands an intelligent mechanism to distinguish whether a packet drop is because of congestion or otherwise.

Last few years have seen many new high speed protocols being presented among which many are the variants of the standard TCP. Among these variants of TCP are: BIC TCP, CUBIC TCP, HS-TCP, H-TCP, TCP Vegas, STCP, CTCP, Fast TCP, LTCP and XCP. Also the mutual comparison of these variants has been a major research interest of last few years. Much research work has been carried on using simulation of these TCP variants. Most of these protocols mainly differ due to the size of the cwnd and the modifications therein. In the standard TCP, whenever there is a packet loss, the size of the cwnd is reduced almost to half, assuming that every time the packet loss has occurred due to congestion in network, and thus sees the only remedy of packet loss to cut the size of congestion window. But this may not always be the case. Packet drop can occur due to many reasons other than network congestion. There are cases when a packet is lost to low signal strength. The packet drop may also occur due to noisy channel. A packet may not reach the destination due to environmental conditions. Thus assuming everything that the packet loss has occurred due to congestion in the network is not logical. This will impose constraints on the network, not allowing the protocol to reach and maintain the maximum transmission rate. This is because the standard TCP follows an additive-increase-multiplicative-decrease strategy.

When a packet loss is detected, it is normally assumed that network is congested and the decrease formula is applied. In case of noisy channel conditions, packet drops are mostly due to errors or noise in the channel which does not indicate network congestion. As most high speed TCP protocols consider every packet drop as an indication of network congestion, this reduces their speed in reaction. This strategy does not allow most protocols to achieve maximum throughput in such network conditions, e.g. CUBIC (a high speed TCP protocol) suffers a lot and often results in its incorrect estimation of the link capacity [15]. Essentially, we need an

intelligent mechanism that can enable these protocols to distinguish between packet drops caused by network congestion and errors in the channel. Indeed this cannot be done always with perfect accuracy but close approximation will truly result in improved performance of these protocols.

In this chapter, first the addressed problem is elaborated and then its implication on high speed protocols. We will also discuss various possible reasons of packet drops. Finally we will talk about our proposed solution in detail in the following chapter.

3.2 Problem Definition

TCP and most of its high speed variants consider every packet loss as an indication of network congestion. On packet drop, irrespective of the reason of the packet drop, they reduce their *cwnd* using their decrease formulae. In a noisy channel conditions particular, this blind reduction of cwnd backfires and restricts protocol to achieve maximum throughput and full bandwidth utilization. Moreover, this approach does not allow the protocol to stabilize, and its cwnd fluctuates a lot disturbing the whole network. Indeed, we need an intelligent mechanism that somehow enables the protocol to distinguish between the drops reason: if it is due to congestion or not. This thesis is aimed at developing such a technique.

3.3 Possible Reasons of Packet Loss

Packet may be lost in the network due to the followings major reasons.

3.3.1 Congestion

A network is said to be congested when a large number of packets are sent to the network which exceeds the network capacity? Here comes the concept of congestion window or *cwnd*. The size of *cwnd* determines the maximum number of packets that can be transmitted at a particular time. The size of *cwnd*, although having an upper bound, is variable whose size is dynamically set by the congestion control algorithm by examining the network. When a sender transmits packets at the rate equal to that upper bound, then the system has the maximum throughput. That is the maximum data rate at which any node can transmit or receive data and no greater transmission rate is possible. Every algorithm starts its *cwnd* with some small initial value, gradually to increase it to the maximum size if there is no congestion in the network assuming that the receiver gets all that the sender transmits. When a sender receives an acknowledgment for the data that is transmitted, it means that the data was successfully received

by destination node, and the sender increases its rate of transmission by increasing the *cwnd* size. Thus for every acknowledgement the sender node goes on increasing its transmission rate until there arrives a time when rate of the transmission exceeds capacity of the network. In such situation different kinds of queues and overflows are made by routers in the network, causing many packets to be lost. That is the 'upper bound' which we mentioned above. Then is the time to reduce the *cwnd*. The TCP algorithm deduces that every packet drop occurs due to congestion in the network without caring for the other factors which may also be the cause of the packet drop.

3.3.2 Receiver Buffer overflows

Receiver buffer overflow happens when a sender sends data packets to the receiver but the receiver, being busy in performing some other applications, is not ready to receive those packets. In this case the sender may overrun the NIC buffer. When there are more loads on the system, the receiver has sufficient space in its window but it is totally unaware of the information that the packets are dropped in its NIC because it is not in active state. The receiver supposes that the packet drops are occurring due to congestion in the network.

3.3.3 Noise

Noise is unwanted signal. It gets added to intended signal on its way. This results in the distortion and disfigurement of the signal in way that it becomes impossible to decode it at the destination end as it was. There are many reasons for noise: electromagnetic interference, crosstalk, network and environment conditions etc. Electromagnetic interference is caused by the electric or magnetic fields of some other nearby electric cables or the electric motor or any other phenomenon that can cause electric or magnetic fields. Crosstalk is the mutual interference of the two wires i.e. the positive and negative one. Cross-talk can be best avoided by grounding the negative wire i.e. setting it to zero voltage. The vulnerability of a signal to noise is determined by its signal-to-noise ration i.e. SNR. When the SNR is high, a signal is less vulnerable to noise.

3.3.4 Channel error

Packets are dropped in wireless network mostly due to error in the channel. As all the channels are shared by all the nodes in the wireless network so there may be greater chance of packet drops.

3.3.5 Collisions

Packet loss may also occur due the collision of two or more packets. For example we have three nodes A, B and C. Both node A and node C are the sender nodes and node B is the destination node. At some time node A sends packet to the node B. At the same time node C also sends packet to the node B not sensed the packet sent by node A. The collision happens after some amount of time. Node A detects a collision when it received the packet of node C and suddenly stops its transmission. Node C detects a collision when it received the packet of node A and suddenly stops its transmission.

3.3.6 Attenuation

Attenuation is the weakening of a signal as it passes through the channel to the destination. On its way to the destination the signal transmitted can get so distorted due to attenuation that it may at times become impossible to regenerate the original signal. Signal amplifiers or regenerative repeaters can be installed after a certain length of transmission lines so that original signal is restored before traveling farther. Attenuation is measured in decibels.

3.4 Effect on High Speed Protocols

TCP and its high speed variants consider every packet loss as an indication of network congestion. On packet drop, without any care for the reason of the packet drop, they reduce their *cwnd* using their decrease formulae. In a noisy channel conditions, this blind reduction of cwnd backfires and restricts protocol to achieve maximum throughput and full bandwidth utilization. Moreover, this approach does not allow the protocol to stabilize and its cwnd fluctuates a lot disturbing the whole network. Indeed we need an intelligent mechanism that somehow enables the protocol to distinguish between the drops reason if it is due to congestion or not. This thesis is aimed at developing such a technique. We have implemented our solution in ns-2. Among high speed transport protocols, we have chosen CUBIC protocol for simulation and analysis, as its performance is badly affected in noisy channel conditions [30], [42].

3.5 CUBIC as a Test Case

As stated earlier, we take CUBIC as a test case. We have used its available implementation at [6]. First we will observe its performance in normal conditions without any

probabilistic drops. Then we will see the effect of randomized packet loss on CUBIC's performance. We will randomly drop packets with certain probability and see its effect on CUBIC. For the increasing probability of packet drop, throughput gets on decreasing. In the following sections, we analyze the performance of CUBIC.

3.6 Performance of CUBIC

Next we simply report the performance of CUBIC in terms of throughput in a network working under normal conditions along with scenario having 0.001 packet drop probability. This means that we have 1 packet randomly dropped in every 1000 packets.

The following graph shows the results of simulation performed in NS-2. We used a simple topology having two nodes (source and destination) connected together with a link having 100 Mbps bandwidth and latency is 64 msec. Simulation time is 50 seconds.

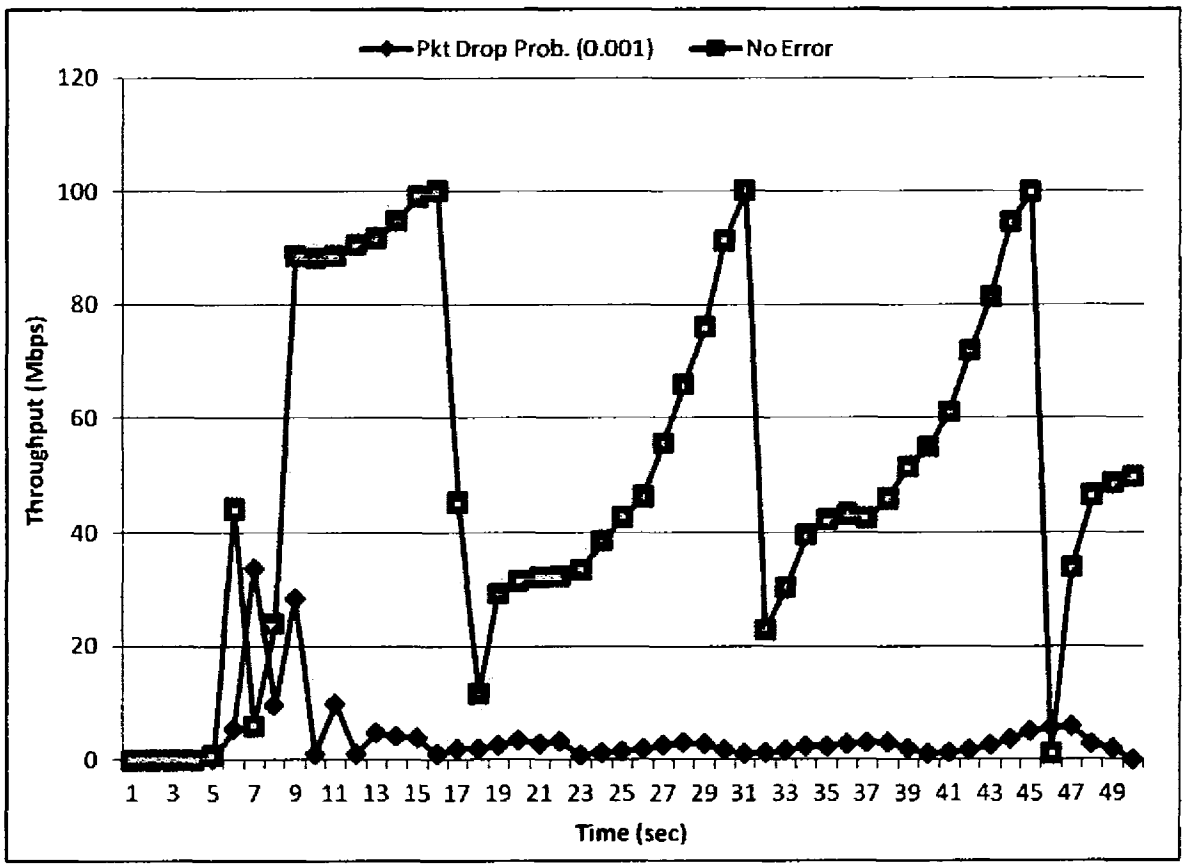

Figure 3.1 Performance of CUBIC

As evident from the graph, CUBIC protocol is able to achieve maximum throughput in normal network conditions at some points (15sec, 30sec and 45sec) and its cwnd is cut down immediately. The average throughput is almost 50 Mbps. However, if we introduce channel error at the rate of 0.001 then its performance is drastically degraded and the average throughput is about 0.5 Mbps. Is this comparable with the 50 Mbps of the previous network?

It should be noted that the same network can perform better under the same conditions or even worse only with the change of algorithm. It can perform far better if it uses the intelligent packet drop estimation strategy, k-NN, as propounded in the thesis.

3.7 Summary

In this chapter the problem is explained briefly. Various reasons of packet loss are introduced and their effect on high speed protocols. Then consider CUBIC as a test case to check its performance both in normal network conditions and random packet drops. We propose our solution to this problem in the following chapter.

Chapter 4

Proposed Solution

PROPOSED SOLUTION

4.1 Introduction

As stated earlier, we take CUBIC as a test case to check the effect of packet drop estimation on the overall transmission process and performance of CUBIC. In the previous chapter, we have seen the performance of CUBIC in normal network conditions. We have also witnessed that performance of CUBIC was greatly degraded when we randomly drop packets with certain probability. For the increasing probability of packet drop, throughput gets on decreasing. Some intelligent solution must be proposed to this problem. One possible solution is to estimate whether the dropped packet was due to congestion or any other reason so as to decide whether or not to reduce the *cwnd*. We use K-nearest neighbor algorithm (k-NN) for such packet drop estimation. The k-NN is a technique which maintains a history of previous instances on the basis of which it estimates whether the packet drop was due to congestion or not. This estimation is done the basis of majority of instances. In the following Section, we briefly explain k-NN algorithm.

4.2 The k-Nearest Neighbor (k-NN) Algorithm

The k-Nearest Neighbor Algorithm, commonly shortened as k-NN, is a classification method that intuitively classifies unlabeled and non-classified objects based on their closest examples from within a feature space of already classified objects. It is a type of instance-based learning, (other examples of instance-based learning being: weighted regression and case-based reasoning). k-NN is one of the simplest algorithms of machine learning.

In k-NN, all instances correspond to points in an n-dimensional Euclidean space. Classification is delayed until the arrival of new instances. Whenever a new instance arrives, it is classified by comparing it with the feature vectors of the different points which are its neighbors, and thus the name k-nearest neighbor. The target function may be discrete or real-valued. This is sensitive to the local structure of the data.

The basic and simple version of k-NN is noted for the ease of implementation. It is done by calculating the distance from the test sample to all the neighboring instances. But this version has vigorous computations, especially for the larger size of k. Over the years, many flavors of k-

NN have been presented. To use the most suitable flavor, makes k-NN computationally tractable even for large values of k.

Example: This is an example of k-NN analysis. Consider the classifications of new query point (which is our instance under consideration) among a number of other know instances within a red circle. What we have to do here is to classify the result of the query point depending on a selected number of its nearest neighbors. Simply put, the task is to find out whether the query point is to be classified as a minus sign or as a plus sign, as shown in the figure.

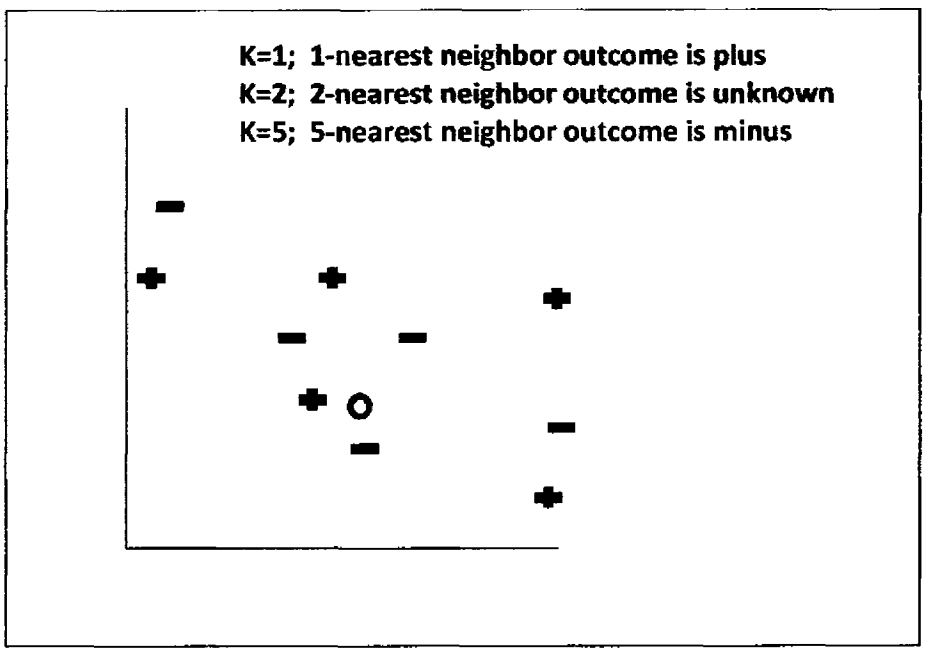

Figure 4.1 k-NN Algorithms

Consider that k=1, that is the k-NN algorithm's outcome is based on 1-nearest neighbor. Then, as it is clear from the figure, the algorithm's outcome will be a plus sign because the closest point to the instance under examination (colored red) is a plus. For k=2, that is 2-nearest neighbor case, the outcome will not be clear as there are two points nearest the red colored instance. As one point is plus and the other minus, thus the result is ambiguous. Now for k=5, there are five neighbors. As clear from the figure, two of them are plus and three minus. Thus the minus signs in majority the instance under consideration is classified as minus. The same can be extended for any number of k-nearest neighbors.

4.3 Proposed Solution

As our proposed solution is based on k-NN, and we have developed a packet drop guesser module considering only two parameters i.e. time and congestion window. Of course, we can have several other parameters but that's not taken into account for this work as the two chosen parameters are considered sufficient and the most relevant to the problem in hand. We get information about network status in the form of two feedback events: a) when we receive an Ack packet, it shows non-congestion as the data is going through. b) When timer expires or Not-Ack packet is received, it shows that network may be congested. So we keep record of these two types of events in our module (i.e. Ack and drop events) and congestion window along with time instant of the event is recorded in the history which will help us in matching a pattern to distinguish a random packet drop caused by noisy channels from genuine packet drops due to network congestion. Obviously, packet drops caused by reasons other than congestion, will be randomly distributed and we will certainly have many ACK events in the neighborhood of such drop events. While packet drops due to congestion will have consistent pattern and congestion window will almost be same for all of them. Thus using simple k-NN technique, we can distinguish packets drops caused by congestion from random packet drops.

Time is the duration for which the simulation is in progress while congestion window determines the number of packets that the system can transmit at that time.

4.4 Design Parameters

In this Section, we explain our different design parameters: Recorded Event Format, Prediction, Maximum Size of Event History, and Range.

Recorded Event format is a record of the events as Drops or Acknowledgements. Due to memory constraints, all the events cannot be recorded. Only a certain number of recently occurred events can record. **Prediction** is done on every drop event. It will use its intelligent packet estimation technique to determine whether the drop was due to congestion or other transmission impairment. The **Maximum Size of Event History** is the number (or size) of events of that we store as the history of events upon which we call the k-NN algorithm for every packet drop. For our simulation it is taken to be 100 events. Whenever the event history gets mature i.e. its size reaches 100 events, on the arrival of new events, the first event is removed.

Range is the Euclidean distance upon the k-NN has to act. Range means the number events in the queue of that we want to compare with the new events to estimate it's reason that how many drops and acknowledgement in the last few event and then to guess whether the current event drop is due to the congestion or any other error. In our simulation it is set to 30.

Below diagram shows how proposed model of the problem works using k-NN.

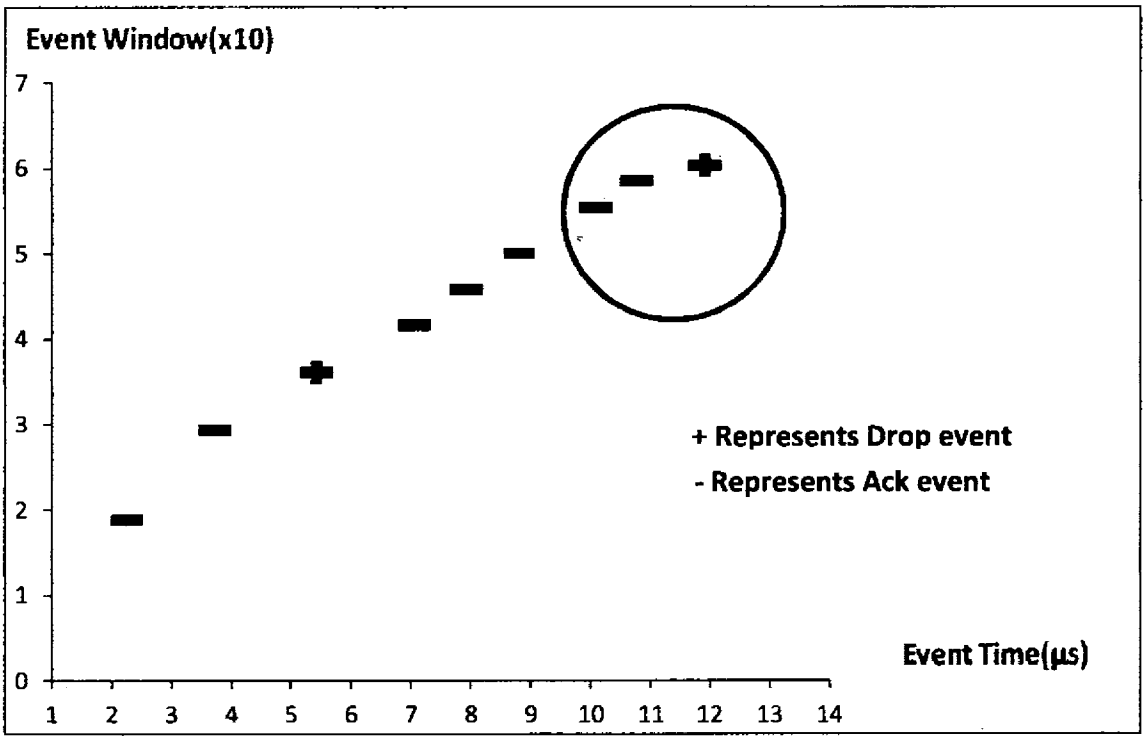

Figure 4.2 Graphical representation of k-NN with parameters

The model is explained with the help of a figure as above. We have Event Window on y-axis and Event Time on x-axis. The graph shows the size of event window size on a particular time.

In the above figure a MINUS sign represents an acknowledgement i.e. No Congestion and a PLUS sign represents a drop i.e. Congestion indication. In the above figure there is an acknowledgement at 2µs (2 micro seconds) and another at about 3µs and then there is a packet drop at about 4.5µs. With this the Event Window at 2 µsec is 20 (2x10) packets and at 3 µsec it is almost 30 packets. Similarly with the packet drop at almost 4.5 µsec, the Event Window is more

than 30 and seemingly less than 33 (the level for the window to be mature enough). This means that all the events are not shown on the graph. Generally we call the prediction function after every event drop. But the congestion window is not mature enough to estimate the correct nature of a drop event. So we have to consider any event as acknowledgement until the size of the event queue goes beyond a certain threshold which for our case is 33. The prediction function will return false for any event if the queue is of size less than 33 based on the pre-mature history (history is not mature). The related chunk of the prediction function is as below.

$$\text{If}\left(\frac{MAXSIZE}{3} > \text{size}\right)$$

return false;

In the above code, MAXSIZE is the maximum possible number of events in the history which we want to maintain. For our case MAXSIZE is 100. By returning false, it means there is no congestion. From above packet drop is seen at about 11μs and at 12^{th} μs the event window is reduced. This is because we consider that the window is now mature and the k-NN algorithm can be implemented upon it. It is clear from the above figure that the algorithm as taken no response for the two drops. But if the process continues then certainly after a few drops there will be decrease in the Event Window's size.

4.5 Proposed Model

The proposed model of the work is shown below.

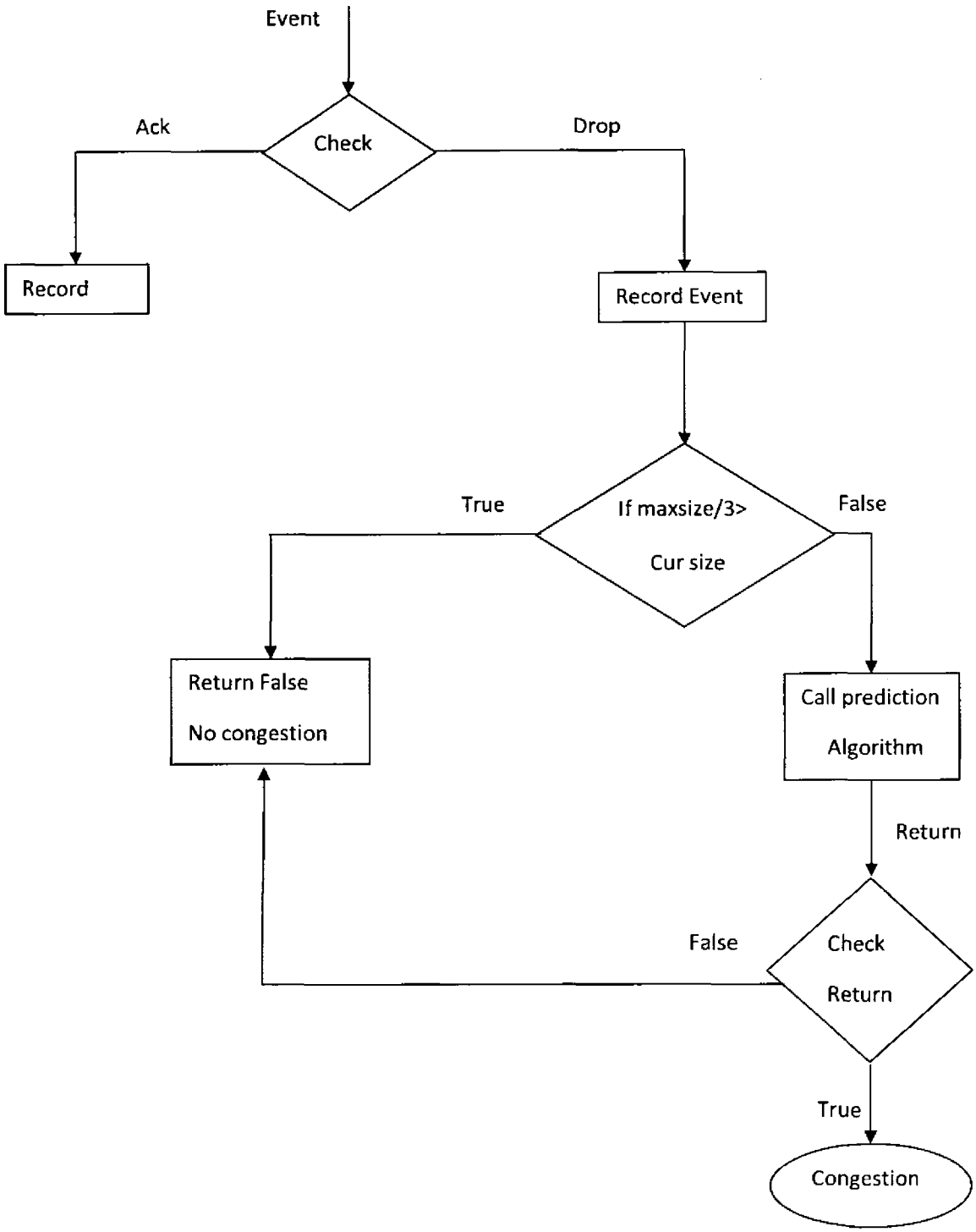

Figure 4.3 Proposed Model

Event Structure

In our event structure, we keep record of three different parameters as shown in following table.

Time	Congestion Window	Event Type

4.6 Algorithm for proposed Technique

The detailed algorithm for our proposed technique is described below.

Packet Drop Guesser Module

Input: Event

Output: Return TRUE/FALSE

Procedure:

1. *If Event_Type= ACK*
    a. *Record Event and Exit.*
2. *Else*
    a. *Record Event*
    b. *If MAX_SIZE/3 > CUR_SIZE*
        i. *Return FALSE // No congestion*
    c. *Else*
        i. *Result= KNN_Prediction_Algorithm()*
        ii. *Return Result*

Our Packet drop guesser module work as follows: Input is event and output is true or false. In first step when the type of the event is Ack then it is simply recorded and exit. But when the type of the event is other than Ack, means Drop then it is recoded and if one-third of the maximum size of the history is greater than the current size then it means there is no congestion and return false otherwise, returns true means congestion. When it returns true, k-NN prediction algorithm is invoked which is given below.

KNN_Prediction_Algorithm

Input: Drop_Event, Event_Record, Range

Output: Return TRUE/FALSE

Procedure:

1. *TRUE_Count=0*
2. *FALSE_Count=0*
3. *For each Event in Event_Record*

   If Event is within Range of Drop_Event

   i. *If Event.Flag = TRUE*

   TRUE_Count++

   ii. *Else*

   FALSE_Count++
4. *End For*
5. *If TRUE_Count < FALSE_Count*

   Return FALSE
6. *Else*

   Return TRUE

4.7 Summary

In this chapter the proposed solution is presented in detail. Also discuss k-NN technique and how it is incorporated in the proposed solution. k-NN algorithm and proposed model was also provided in this section. The simulation, analysis and comparison of results will be discussed in the following chapter.

Chapter 5

Simulations, Analysis and Comparison of Results

SIMULATIONS, ANALYSIS AND COMPARISON OF RESULTS

5.1 Network Simulator (NS-2)

NS-2 is popular network simulator which was developed by VINT (Virtual Interchange Test Bed). NS-2 is applicable to discrete events. Various protocols are used by NS-2 for the simulation of wired and wireless networks.

Now we explain the way in which the arrangements are made for running a simulation. First of all, a program is written in OTcl script. Then event scheduler is initiated by this script, and exploiting the network objects, network topology is arranged. Then functions are plumbed in the library and the trace resources are guided about the start and stop timing by the even scheduler.

Owing to this we can regard the NS-2 an OTcl interpreter. NS-2 is an OTcl interpreter as reads/decodes the script written in OTcl. After this, environment parameters are set up by NS-2.

For a programmer writing an OTcl script, it will be easier to make compound objects from the object library.

At the end of the simulation, all the related data is outputted and stored in text files. These text files are very important. Their importance can hardly be exaggerated as they contain data of all the simulation. Any result can be extracted from these text files. They offer a great ease in judging the overall result of a simulation. These files are taken as input by NAM – a graphical user interface, standing for Network Animator. NAM plots the text files' results on graphs and gives great ease to assess the simulation.

C++ can also be incorporated into the OTcl scripts which are used by NS-2, which offer great ease in certain aspects. The event scheduler is written and compiled using C++. C++ is used for defining other components too in the data path. This reduces the event processing time of the packets.

A UNIX environment is needed for running NS-2 simulations. NS-2 can also be run on windows environments: Cygwin should be used. Cygwin is software that virtually provides the required UNIX environment. Physical activities get translated into events in NS-2. Thus a discrete even simulator is needed. These discrete events are later processed for their corresponding scheduled occurrences. The events are processed with the passage of time. The

simulation time may not necessarily be the real life time.

5.2 Simulations parameters

All the simulations are performed in NS-2 using TCP/Linux code available at [25]. We have incorporated our module into CUBIC protocol as a test case. CUBIC was selected because its performance drastically degraded when random packet drops are introduced [6]. Secondly here we used simple topology consisting of two nodes called source node X and destination node Y. The link used between these two nodes is 100 Mbps.

Figure 5.1 Simple Topology

The simulation parameters are shown in the table below.

No. of Flows	1
link Delay	64msec
Packet size	1448 bytes
Buffer size	220 packets
Simulations time	50 sec
Minimum Bandwidth	100 Mbps

Table 5.1 Simulations Parameters

5.3 Performance parameters

Different protocols can be compared and contrasted on the basis of certain parameters. For our case these parameters are throughput, link utilization, and congestion window. Below we give brief description of each of them.

- **Throughput:** Throughput is the main performance parameter of our study. It tells how close to success our results are. It is the rate at which a device sends successfully and can be expressed in term of Mbps.

$$\text{Throughput} = \frac{\text{Total Data Delivered}}{\text{Simulation Time}}$$

- **Link utilization:** It is the percentage of the bandwidth of the link that is currently being used. For the link utilization to be more, the better.

$$\text{Link Utilization} = \frac{\text{Throughput}}{\text{Bandwidth}} * 100\%$$

- **Congestion window:** Shortened as *cwnd*, the congestion window determines the maximum number of packets that can be transmitted at a particular time. The congestion window is a value read at particular time from CUBIC protocol. The size of the congestion window is however a variable that gets reduced when the network is noisy, as in our case.

5.4 Results

Now we provide the results of our work. The first three graphs show the results of the throughput, congestion window and link utilization of the CUBIC protocol when there is no random packet drop.

The results also include the final statistics for the CUBIC algorithm with k-NN technique for the features: throughput, link utilization and congestion window. For throughput the results are from three different environments with packet drops rate of 0.1, 0.01, and 0.001 randomly. The same is also done with link utilization and congestion window but these results were very weak, We have presented their averaged values here.

5.4.1 Throughput with No Random Packet Drops in CUBIC

The result of the throughput is shown in the figure below. Actually there is no random packet drops in this case. So the throughput of CUBIC protocol is much better. At 4 second CUBIC protocol has unexpected behaviors due the transient state. After 4 second it reaches the maximum throughput at 5, 15, 30 and 45 second. When the flow reaches the maximum, it cuts down the window immediately.

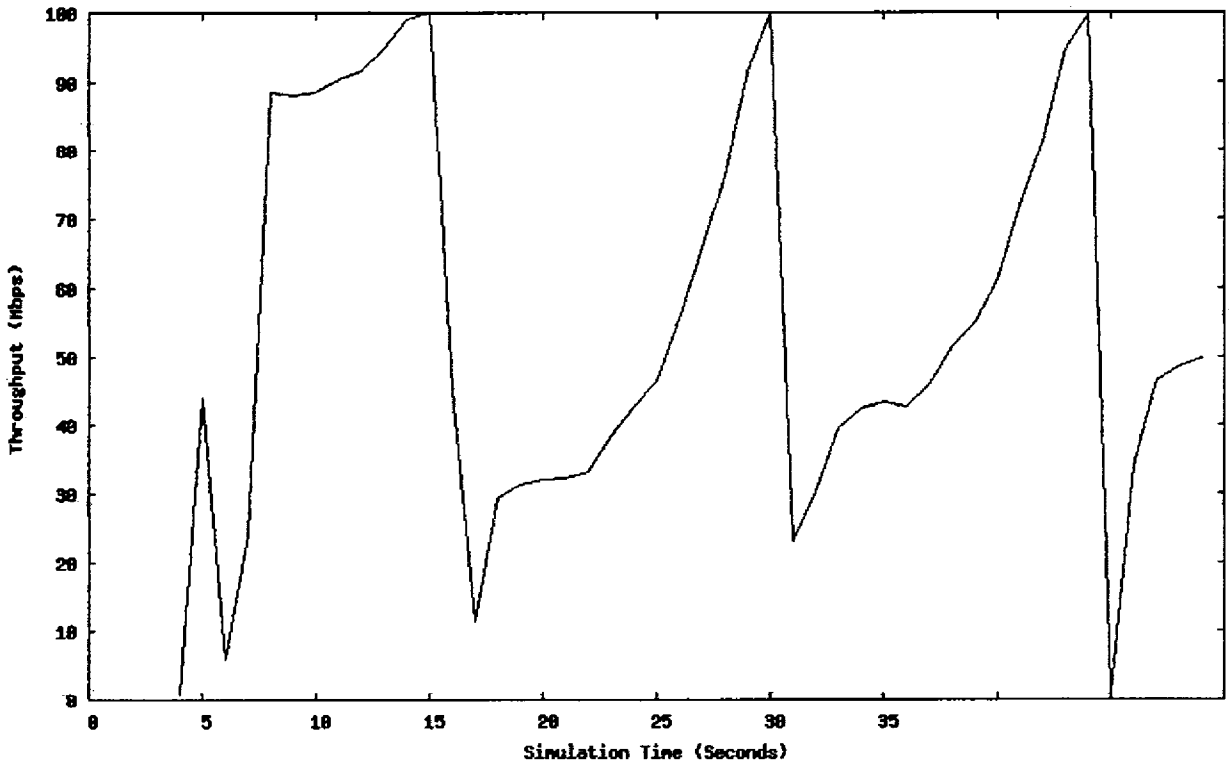

Figure 5.2 Simulation of throughput with no random packet drops in CUBIC

5.4.2 Congestion window with No Random Packet Drops in CUBIC

The result of the congestion window is shown in the figure below. Actually there is no random packet drops in this case. So the congestion window of CUBIC protocol is better. In transient state the CUBIC protocol make wrong estimation. Congestion reaches the maximum size which is 100 packets at 5, 15, 30 and 45 second. By reaching the maximum size the window is cuts down to zero and start again.

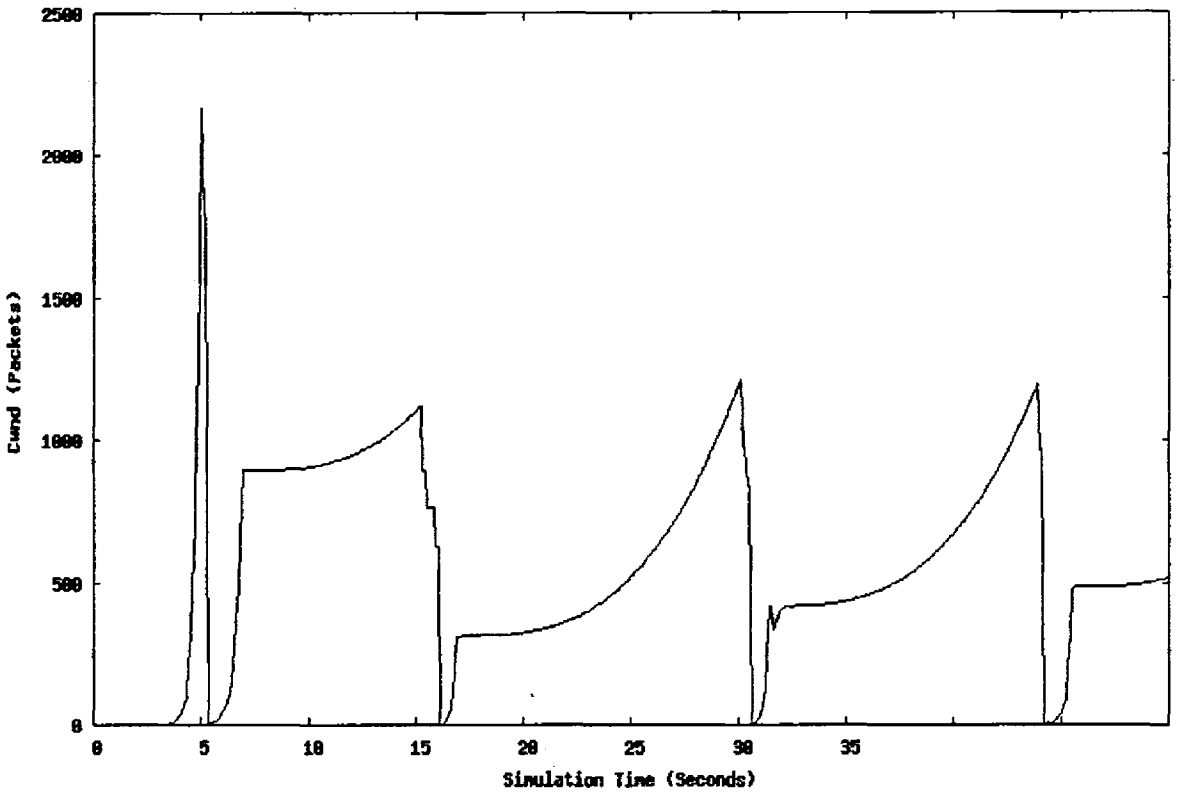

Figure 5.3 Simulation of congestion window with no random packet drops in CUBIC

5.4.3 Link utilization with No Random Packet Drops in CUBIC

The link utilization is shown in the figure below. The link utilization of CUBIC protocol is better in the case because there is no random packet drops. CUBIC has unexpected behavior at the start. But after 5 second, link is utilized (percentage of the bandwidth of the link that currently being used) in efficient manner by reaching to 100 % at time 5, 10, 30 and 45 second. Reaching to the target point, CUBIC protocol flow immediately fall down to almost zero and start again in the search of maximum link utilization.

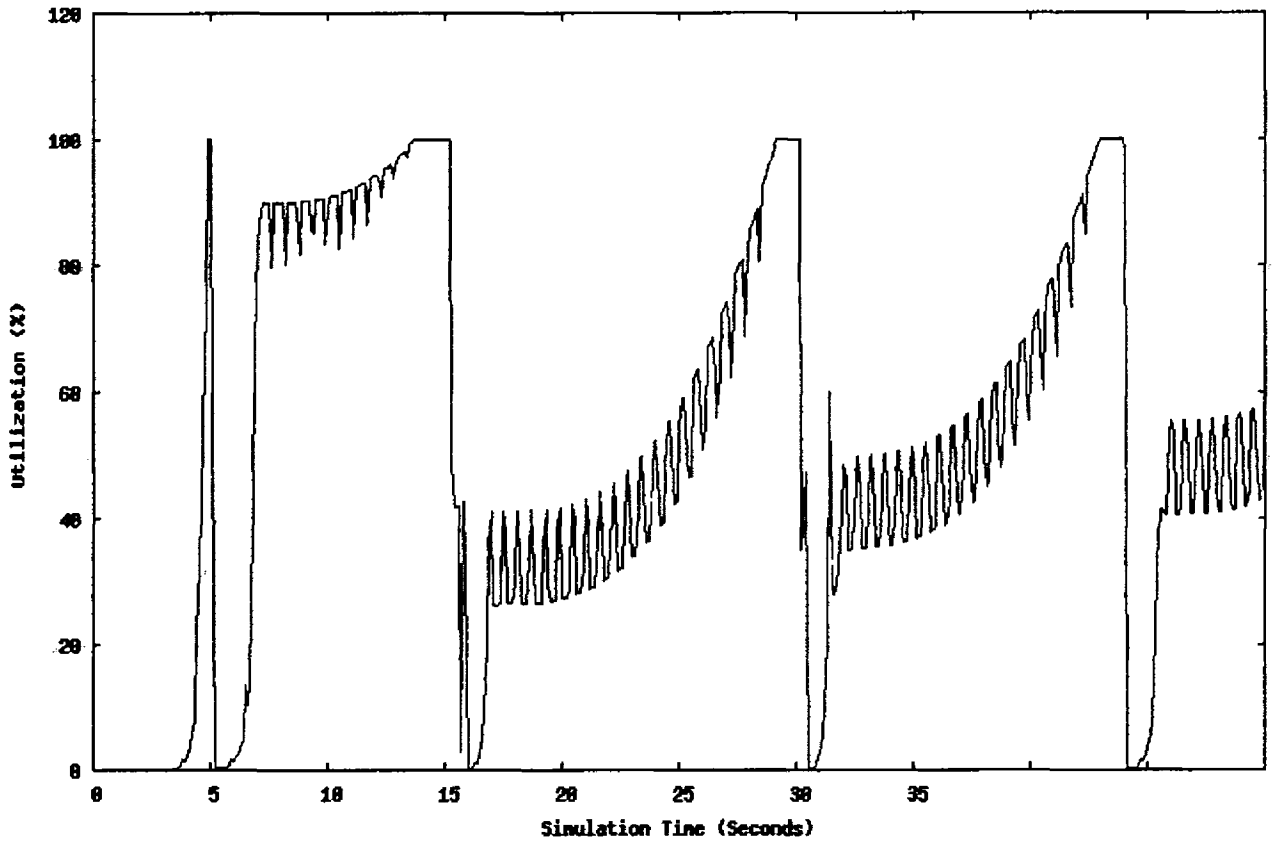

Figure 5.4 Simulation of link utilization with no random packet drops in CUBIC

5.4.4 Throughput (Random packet drops rate is 0.1)

The figure below shows the throughput when packet drops rate is 0.1 randomly. An error rate of 0.1 means that there is 1 packet dropped out of every ten packets (needless to say this is an extremely high error rate). Whenever there is no error (normal CUBIC) in the network, throughput is better as shown. Shown in the below graph are results of the CUBIC without k-NN (shown in blue) which as expected and as it should be are almost zero at every point. Next is the results of CUBIC with k-NN technique (shown in green), these are obviously better than the CUBIC without k-NN results and almost reaching an average throughput of 20 Mbps. These values are very sharp and to present a smoother graph moving average of 5 previous values is also shown (in the black).

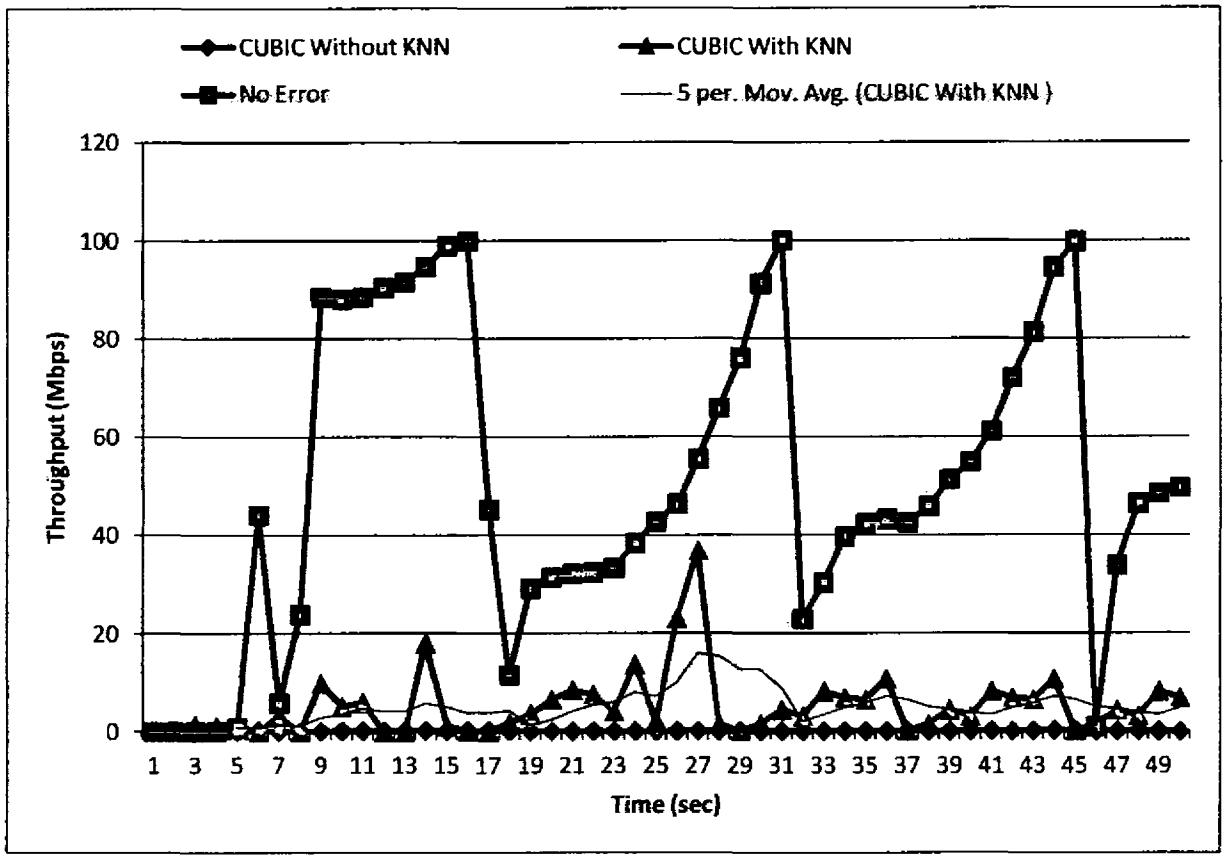

Figure 5.5 Throughput when random packet drops rate is 0.1

5.4.5 Throughput (Random packet drops rate is 0.01)

The figure below shows the throughput when error rate is 0.01. It means that there is 1 packet dropped out of every hundred packets, which is comparably stable environment than the one with an error rate of 0.1. Whenever there is no error (normal CUBIC) in the network, then throughput is better as shown. But when the error occurs at rate of 0.01, then the throughput is affected almost comes down to 0 Mbps. So we incorporate k-NN module to achieve some acceptable level of throughput which is an average of 30 Mbps.

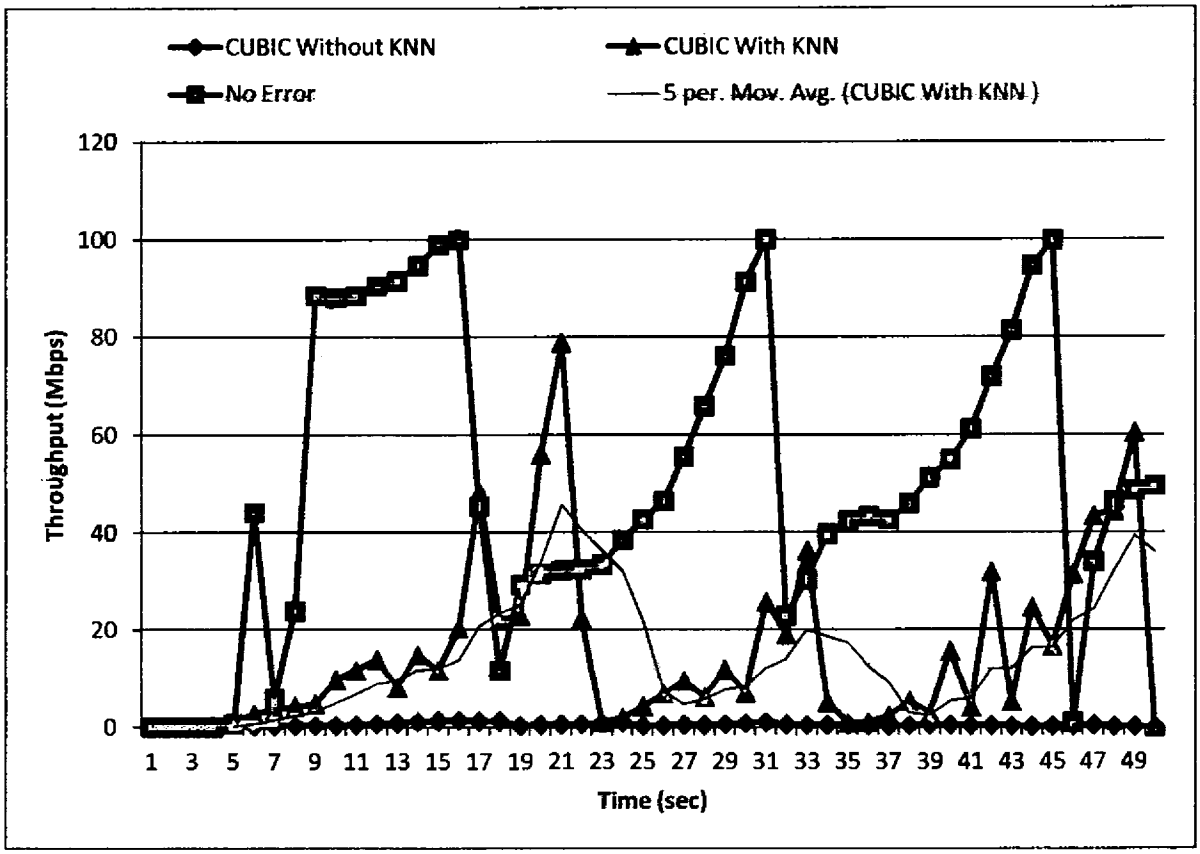

Figure 5.6 Throughput with random packet drops rate is 0.01

5.4.6 Throughput (Random packet drops rate is 0.001)

The figure below shows the throughput when error rate is 0.001. It means that there is 1 packet dropped out of every one thousand packets. Whenever there is no error (normal CUBIC) in the network, then throughput is very fine as shown. But when the error occurs at rate of 0.001, then the throughput is deteriorated badly and is slightly greater than 5 Mbps. So we incorporate k-NN module to achieve some acceptable level of throughput which is almost 35 Mbps. So the throughput gets better and better when the random packet drop rate is decreased.

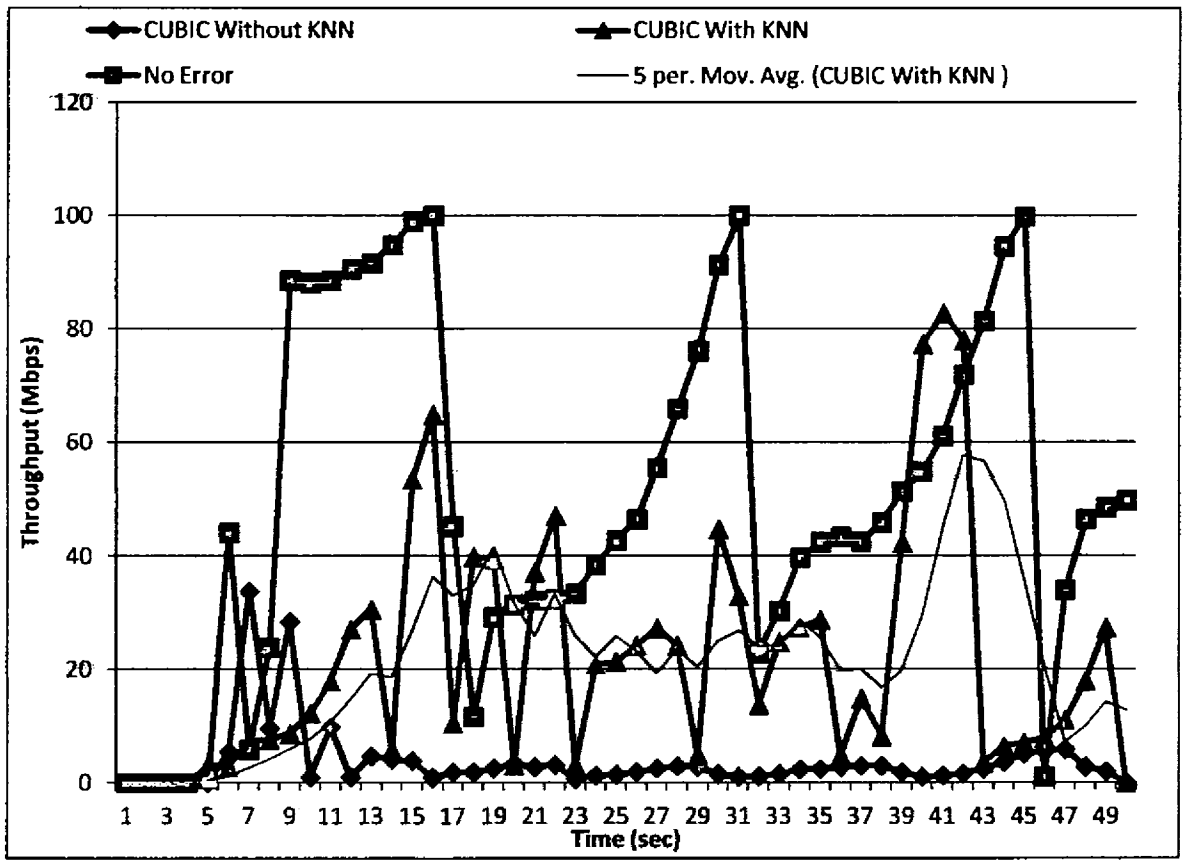

Figure 5.7 Throughput when random packet drops rate is 0.001

5.4.7 Average Congestion Window

The figure below shows average congestion windows for all the three instead of different graphs of every environment separately. The throughput graph was shown with the time scale on the x-axis while here we are not doing so, instead only the average value at any time is shown for each of the environment. For the packet dropping probability of 0.1, without k-NN module the congestion window is approximately 2 packets. But by incorporating the k-NN module the congestion window is increased and jumps to approximately 10 packets. When the packet drops rate is 0.01 the congestion window without k-NN is increase to 7 packets. After incorporating k-NN module the performance is improved and now congestion window is increased to average of 20 packets. But in case of 0.001 packets drops rate the congestion window without k-NN reaching to 10 packets and with k-NN it is further improved and jumps to an average of 40 packets.

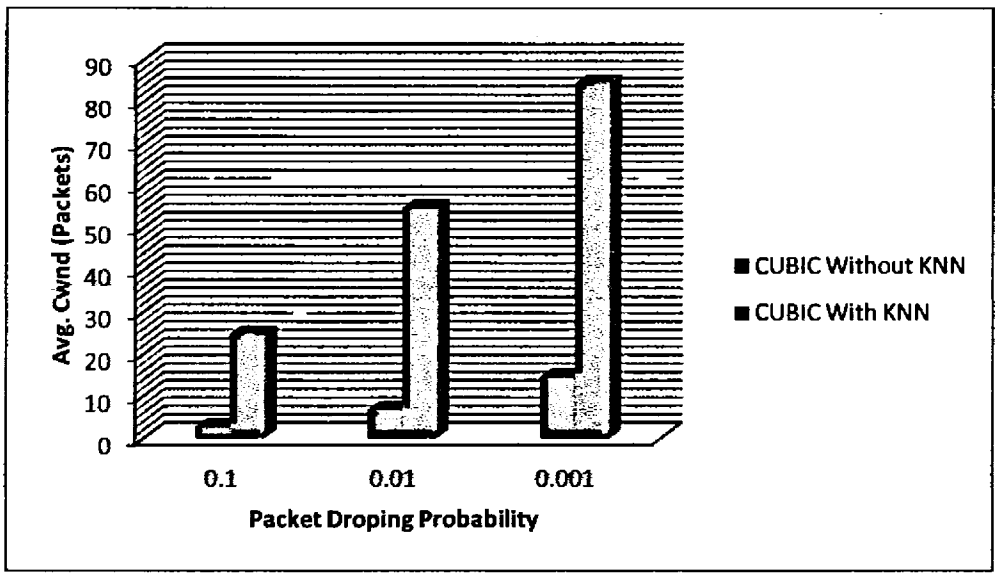

Figure 5.8 Effect on Avg. Congestion window under different packet dropping probability rate

5.4.8 Average Link Utilization

In the figure below, we have shown average link utilization for all the three environments instead of different graphs of each environment separately. The throughput graph was shown with the time scale on the x-axis while here we are not doing so, instead only the average value at any time is shown for each of the environment. The average link utilization is about 2 % when the packet drop is 0.1 without k-NN module. When k-NN module is employed the link utilization is about 7 %. When the packet drops rate is 0.01 then the link utilization for without k-NN is 3 % and with k-NN it is improved to 20 %. But in case of 0.001 packet drops rate, the link utilization for without k-NN is 4 % and for with k-NN it is 40 %.

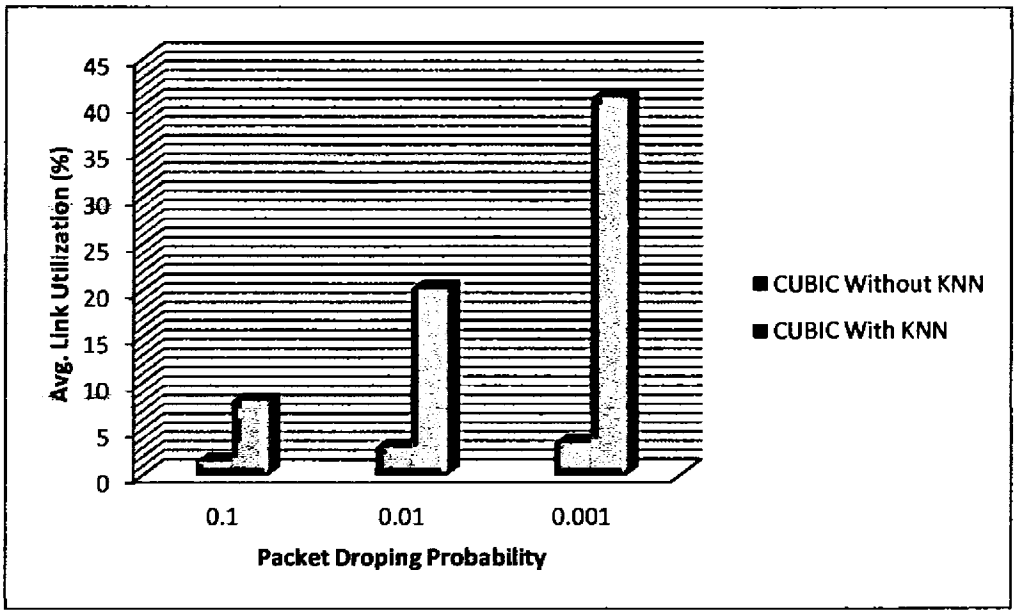

Figure 5.9 Effect on Avg. link utilization under different packet dropping probability rate

5.5 Summary

In this chapter first Network simulator NS-2 was introduced. Then simple network topology with different simulations parameters was discussed. Also different results were shown in graphs and were compared that how it is worked without k-NN and with k-NN module. The performance with k-NN is found better than without k-NN in random packet drops rate. The next chapter contains conclusion and future works.

Chapter 6

Conclusion and Future works

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

The performance of the CUBIC algorithm, though fine under normal conditions, deteriorates badly under noisy conditions. The performance of CUBIC algorithm in an environment with no errors and the results are seen for its throughput, congestion window and link utilization. It is found that the performance of CUBIC in this environment was very well and comparable to any other high-speed network algorithm. The same procedure was done in a very noisy environment where the error rate was 0.1, 0.01 and 0.001 and its performance is degraded. So CUBIC's performance was gradually becoming better and better as the environment became less and less noisy and the error rate went on decreasing. The k-NN module, when incorporated into the CUBIC algorithm, considerably rectifies its performance. There is a flaw in the k-NN module in CUBIC: it has late response to the negative acknowledgements. On the first packet drop, it takes no action and takes it an outcome of some channel error. It does the same for the next few event drops and when confirmed that there is some sort of congestion in the channel, it reduces its congestion window size. Its performance could be considerably better if it does not wait for more than few events and takes immediate steps. However, this was not possible and this drawback was to be traded for the chances that most of the packet drops are single and not successions of them.

6.2 Future Works

Now that we are done with our thesis, much has been achieved but there is no end to it. There is still a long way to go. Still more modifications and extensions can be made so that we get a better and better protocol. The first and foremost of all these is to undertake a comprehensive analysis and to critically evaluate the performance of the CUBIC with k-NN incorporation. We have evaluated only three of the features (i.e. throughput, link utilization and congestion window) under the range of 30 events. Further work can include many other features with different environments and ranges. Also, as shown in the results and again summed up in the conclusion, there is flaw with the k-NN technique that does not provide immediate feedback so the congestion can be updated quite in time for better performance. In this regards, the further

work will be to provide immediate feedback about congestion points to our k-NN module. This will enable the protocol to achieve the desired performance. And one other future task will be to incorporate this k-NN in the other algorithms as well. Here the k-NN technique is incorporated in CUBIC only and its performance was greatly improved. The same done with other high-speed network protocols will also improve their qualities.

References

REFERENCES

[1]. L.G Roberts and B.D. Wessler , Computer Communication networks, N. Aramson and F.F.Kuo Proc of Fips,pp.281, 1973.

[2]. Murthy, C.S.R. & Manoj , Adhoc Wireless sensor networks, architecture and protocols, printice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[3]. Junsoo Lee, Stephan Bohacek, Jo˜ao P. Hespanha, Katia Obraczka "A Study of TCP Fairness in High-Speed Networks", University of Southern California, white paper 2007.

[4]. S. Molnár, B. Sonkoly, and T. A. Trinh, "A comprehensive TCP fairness analysis in high speed networks," *Computer Communications*, vol. 32, no. 13-14, pp. 1460–1484, Aug. 2009.

[5]. Tom Kelly "scalable TCP: Improving performance in Highspeed Wide Area Networks , 2004.

[6]. K.Satyanarayan reddy and Lokanatha C.Reddy " A Survey on Congestion Control Protocols For high Speed Networks" IJCSNS International Journal of Computer Science and Network Security, Vol.8 No.7 July ,2008.

[7]. L. XU, K.Harfoush, and I.Rhee, "Binary Increasse Congestion Control for fast long distance Networks", In Proceeding of the IEEE INFOCOM, Hong Kong, March,2004.

[8]. Injong Rhee, and Lisong Xu "CUBIC: A New TCP-Friendly High-Speed TCP Variant" Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534 USA, 2006.

[9]. M.Jehan, G.Radhamani and T.Kalakumari "Experimental Evaluation of TCP BIC and Vegas in MANETs" International Journal of Computer Applications (0975-8887) Volume 16-No.1, 2010.

[10]. Shashank Jain and Gaurav Raina "An experimental evaluation of CUBIC TCP in a small buffer regime" 2010.

[11]. S. Ha, I. Rhee, and L. Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, 2008.

[12]. Jiawei Chen "Congestion Control Mechanisms of transport Layer Protocols" Helsinki University of Technology, 2008.

[13]. Pankaj Sharma, "Performance Analysis of High-Speed Transport Control Protocols", Master of Science (Computer Science) Thesis, Clemson University, August 2006.

[14]. C. Jin, D. X. Wei, and S. H. Low, Hegde, S. FAST TCP: Motivation, Architecture, Algorithms, and Performance". IEEE/ACM Transactions on Networking, Volume 14, Issue 6, pp. :1246 – 1259, Dec. 2006.

[15]. Cao Yuan, Liansheng Tan, Lachlan L.H.Andrew, Wei Zhang, and Moshe Zukerman "A Generalized FAST TCP Scheme",2008.

[16]. S. Floyd. "Highspeed TCP for large congestion window", Internet Draft draft-floyd-tcp-highspeed-01.txt, February 2003.

[17]. Michele C. Weigle, Pankaj Sharma, and Jesse R. Freeman IV, "Performance of Competing High-Speed TCP Flows" Networking 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems, pp. 476-487, 2006.

[18]. E. Altman, K. E. Avrachenkov, B. J. Prabhu "Fairness in MIMD Congestion Control Algorithms", In Proceedings of the IEEE INFOCOM, 2005.

[19]. Xiuchao Wu "A Simulation Study of Compound TCP" School of Computing, National University of Singapore Computing 1, Law Link, Singapore July 14, 2008.

[20]. Tom Kelly "Scalable TCP: Improving performance in High speed Wide Area Networks" 2005.

[21]. M.Jehan , Dr.G.adhamani, Scalable TCP: Better throughput in TCP Congestion Control Algorithms On MANET's", 2010.

[22]. Habibullah Jamal, Kiran Sultan" Performance Analysis of TCP Congestion Control Algorithms". INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS, Issue 1, Volume 2, 2008.

[23]. Sumitha Bhandarkar, Saurabh Jain and A. L. Narasimha Reddy, "Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control", International Workshop on Protocols for Fast Long- Distance Networks, February 2005.

[24]. D. M. Lopez-Pacheco and C. Pham "Robust Transport Protocol for Dynamic High-Speed Networks: enhancing the XCP approach" in Proc. of IEEE MICCICON, 2005.

[25]. http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/.

[26]. Cao Yuan, Liansheng Tan, Lachlan L.H.Andrew, Wei Zhang, and Moshe Zukerman "A Generalized FAST TCP Scheme", 2008.