Analysis of TCP Variants Using Network Simulator and Emulator on Linux Platform

Harshit B. Parmar[1], Kapil S. Raviya[2]
[1]*BE Student, Department of Electronics & Communication Engineering, C.U.Shah College of Engineering and Technology, Wadhwan, Gujarat, India*
[2]*Assistant Professor, Department of Electronics & Communication Engineering, C.U.Shah College of Engineering and Technology, Wadhwan, Gujarat, India*

Abstract—TCP/IP is the main and most widely used transport protocol for reliable communication since the Internet arrived. The use of original TCP/IP protocol on wireless links in spreading the Internet has encountered some serious performance issues. New versions of the TCP have been proposed to improve data transmission performance. Because of its widespread need, researchers have been studying and proposing new TCP variants trying to improve its behavior. There are different variants of TCP named TAHOE, RENO, NEW RENO, TCP VEGAS, COMPOUND TCP, HS (High Speed) TCP etc. Network Simulation and Emulation are widely used to Develop, Test and Debug new protocols, to explore and study a specific Network related research issue, or to evaluate the network performance of an existing protocol. In this paper, we will compare the performance of different TCP variants in terms of different parameters and the simulation results can provide further insight into the advantages and drawbacks of TCP variants.

Keywords - Congestion window, Corruption, Throughput, Fast retransmit, Fast recovery, Slow Start Threshold, NS2, RTT, DUPACK.

I. INTRODUCTION

Internet traffic is basically made of up of small data bursts called packets. These packets contain information about the origin and destination of the data. The packets are created and reassembled by the transmission control protocol and sent over the Internet by the Internet protocol.

Originally, TCP was developed for wired links and wired links have very less chances of high delay and data corruption due to external parameters. Congestion is the main reason of packet loss on wired links. So, TCP was designed by keeping in mind the above parameters. Wireless links have several problem of variable and high delay with high Bit Error Rate (BER). So initially, unmodified old TCP started to perform badly on wireless links. To deal with the problems of wireless links, a research started in the field of TCP and modifications were done according to the requirements to improve the performance. Variants named Tahoe, Reno, New Reno and SACK and many more came into existence. TCP is a reliable connection oriented end-to-end protocol. TCP ensures reliability by starting a timer whenever it sends a segment. If it does not receive an acknowledgement from the receiver within the „time-out" interval then it retransmits the segment. We shall take brief look at each of the congestion avoidance algorithms and see how they differ from one another.

II. BASICS OF TCP VARIANTS

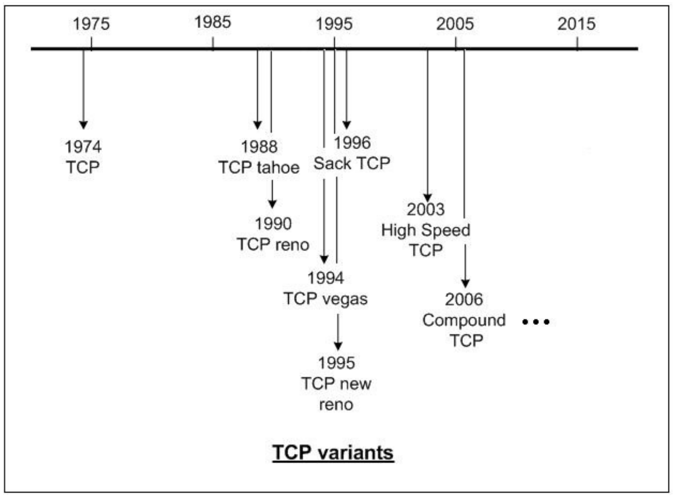


Fig.1: Evolution of TCP Variants

A. TCP Tahoe

TCP Tahoe, one of the variant of TCP congestion control algorithm, was suggested by Van Jacobson in his paper and added some new improvement on the TCP completion in the early stage. That enhance consists congestion avoidance, slow start and fast retransmission.

Tahoe detects packet losses by timeouts and then retransmit the lost packets. A packet loss is taken as a sign of congestion and Tahoe saves the half of the current window as ssthresh value. Then it set cwnd to one and starts slow start until it reaches the threshold value. After that it increments linearly until it encounters a packet loss. Thus it increase it window slowly as it approaches the bandwidth capacity.

Limitations of Tahoe:

The problem with Tahoe is that it takes a complete timeout interval to detect a packet loss. In fact, in most implementations it takes even longer. Also since it doesn't send immediate ACK's, it sends cumulative acknowledgements, therefore it follows a 'go back n' approach. Thus every time a packet is lost it waits for a timeout and the pipeline is emptied. This offers a major cost in high band-width delay product links. TCP Tahoe does not deal well with multiple packet drops within a single window of data.

B. TCP Reno

Reno is defined as a TCP containing the slow start, fast retransmit, fast recovery and congestion avoidance algorithms. This Reno retains the basic principle of Tahoe, such as slow starts and the coarse grain re-transmit timer. However it adds some intelligence over it so that lost packets are detected earlier and the communication path (pipeline) is not emptied every time a packet is lost. Reno requires that we receive immediate acknowledgement whenever a segment is received. The logic behind this is that whenever we receive a duplicate acknowledgment, then his duplicate acknowledgment could have been received if the next segment in sequence expected, has been delayed in the network and the segments reached there out of order or else that the packet is lost.

If we receive a number of duplicate acknowledgements then that means that sufficient time have passed and even if the segment had taken a longer path, it should have gotten to the receiver by now. There is a very high probability that it was lost. So Reno suggests an algorithm called 'Fast Re-Transmit'.

Problems:

Reno performs very well over TCP when the packet losses are small. But when we have multiple packet losses in one window then RENO doesn't perform too well and it's performance is almost the same as Tahoe under conditions of high packet loss. The reason is that it can only detect a single packet loss. If there is multiple packet drops then the first info about the packet loss comes when we receive the duplicate ACK's. But the information about the second packet which was lost will come only after the ACK for the retransmitted first segment reaches the sender after one RTT.

C. TCP Vegas

TCP Vegas is a modified version of TCP Reno. It builds on the fact that proactive measure to encounter congestion is much more efficient than reactive ones. It tried to get around the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule. Also it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and it also suggests a modified slow start algorithm which prevents it from congesting the network. It does not depend solely on packet loss as a sign of congestion. It detects congestion before the packet losses occur. However it still retains the other mechanism of Reno and Tahoe, and a packet loss can still be detected by the coarse grain timeout of the other mechanisms fail. The three major changes induced by Vegas are:

- New Re-Transmission Mechanism:

Vegas extend on the re-transmission mechanism of Reno. It keeps track of when each segment was sent and it also calculates an estimate of the RTT by keeping track of how long it takes for the acknowledgment to get back. Whenever a duplicate acknowledgement is received it checks to see if the (current time segment transmission time)> RTT estimate; if it is then it immediately retransmits the segment without waiting for 3 duplicate acknowledgements or a coarse timeout. Thus it gets around the problem faced by Reno of not being able to detect lost packets when it had a small window and it didn't receive enough duplicate Ack's. To catch any other segments that may have been lost prior to the retransmission, when a non-duplicate acknowledgment is received, if it is the first or second one after a fresh acknowledgement then it again checks the timeout values and if the segment time since it was sent exceeds the timeout value then it re-transmits the segment without waiting for a duplicate acknowledgment. Thus in this way Vegas can detect multiple packet losses. Also it only reduces its window if the re-transmitted segment was sent after the last decrease. Thus it also overcome Reno's shortcoming of reducing the congestion window multiple time when multiple packets are lost.

- Congestion avoidance:

TCP Vegas is different from all the other implementation in its behavior during congestion avoidance. It does not use the loss of segment to signal that there is congestion. It determines congestion by a decrease in sending rate as compared to the expected rate, as result of large queues building up in the routers. Thus whenever the calculated rate is too far away from the expected rate it increases transmissions to make use of the available bandwidth, whenever the calculated rate comes too close to the expected value it decreases its transmission to prevent over saturating the bandwidth. Thus Vegas combats congestion quite effectively and doesn't waste bandwidth by transmitting at too high a data rate and creating congestion and then cutting back, which the other algorithms do.

- Modified Slow-start:

TCP Vegas differs from the other algorithms during its slow-start phase. The reason for this modification is that when a connection first starts it has no idea of the available bandwidth and it is possible that during exponential increase it over shoots the bandwidth by a big amount and thus induces congestion. To this end Vegas increases exponentially only every other RTT, between that it calculates the actual sending through put to the expected and when the difference goes above a certain threshold it exits slow start and enters the congestion avoidance phase.

Problems:

TCP Vegas has the amazing property of rate stabilization in a steady state, which can significantly improve the overall throughput of a TCP flow. Unfortunately, later research discovered a number of issues, including underestimates available network resources in some environments (e.g. in the case of multipath routing) and has a bias to new streams (i.e. newcomers get a bigger share) due to inaccurate RTTmin estimates.

D. TCP New Reno

New RENO is a modified version of TCP RENO. New RENO is able to detect multiple packet losses. Thus it's much more efficient than RENO in the event of multiple packet losses. New-Reno also enters into fast-retransmit when it receives multiple duplicate packets like RENO, however it differs from RENO in that it doesn't exit fast-recovery until all the data which was out standing at the time it entered fast recovery is acknowledged. Thus it overcomes the problem faced by Reno of reducing the CWD multiples times. The fast-transmit phase is same as in Reno. The difference in the fast recovery phase which allows for multiple re-transmissions in new-Reno. Whenever new-Reno enters fast recovery it notes the maximums segment which is outstanding. The fast-recovery phase proceeds as in Reno, however when a fresh ACK is received then there are two cases:

- If it ACK's all the segments which were outstanding when we entered fast recovery then it exits fast recovery and sets CWD to ssthresh and continues congestion avoidance like Tahoe.
- If the ACK is a partial ACK then it deduces that the next segment in line was lost and it re-transmits that segment and sets the number of duplicate ACKS received to zero. It exits Fast recovery when all the data in the window is acknowledged.

Problems:

New-Reno suffers from the fact that it takes one RTT to detect each packet loss. When the ACK for the first retransmitted segment is received only then can we deduce which other segment was lost.

E. TCP Sack

Fast retransmission and fast recovery can only handle one packet loss from one window of data. TCP may experience poor performance when multiple packets are lost in one window. To overcome this limitation, recently the selective acknowledgement option (SACK) is suggested as an addition to the standard TCP implementation.

TCP with "Selective Acknowledgments" is an extension of TCP Reno and it works around the problems face by TCP RENO and TCP New-Reno, namely detection of multiple lost packets, and re-transmission of more than one lost packet per RTT. SACK retains the slow-start and fast retransmits parts of RENO. It also has the coarse grained timeout of Tahoe to fall back on, in case a packet loss is not detected by the modified algorithm. SACK TCP requires that segments not be acknowledged cumulatively but should be acknowledged selectively. Thus each ACK has a block which describes which segments are being acknowledged. Thus the sender has a picture of which segments have been acknowledged and which are still outstanding. Whenever the sender enters fast recovery, it initializes a variable pipe which is an estimate of how much data is outstanding in the network, and it also set cwnd to half the current size. Every time it receives an ACK it reduces the pipe by 1 and every time it retransmits a segment it increments it by 1. Whenever the pipe goes smaller than the cwnd window it checks which segments are un received and send them. If there are no such segments outstanding then it sends a new packet. Thus more than one lost segment can be sent in one RTT.

Problems:

The biggest problem with SACK is that currently selective acknowledgements are not provided by the receiver to implement SACK we'll need to implement selective acknowledgment which is not a very easy task.

F. High-speed TCP (HSTCP)

Because High-speed TCP's modified response function would only take effect with higher congestion windows, High-speed TCP does not modify TCP behavior in environments with heavy congestion, and therefore does not introduce any new dangers of congestion collapse. High-speed TCP improves the performance of TCP in high-bandwidth environments. High-speed TCP (HSTCP)deserves special attention as it is engineered to behave along a similar response function as standard TCP, hut scaled up to meet the demands of higher window sire at higher path loss probability

High Speed TCP (HSTCP) is a modification proposed by S. Floyd to the TCP response function in order to acquire faster the available bandwidth (and faster reach full utilization of the link) in high bandwidth-delay product networks. The targeted network environments for HSTCP are low packet loss rate networks, therefore HSTCP proposes a faster congestion window increase compared to TCP. High Speed TCP's modified response function only takes effect with higher congestion window. It does not modify TCP behavior in environments with heavy congestion, and therefore does not introduce any new dangers of congestion collapse.

G. Compound TCP

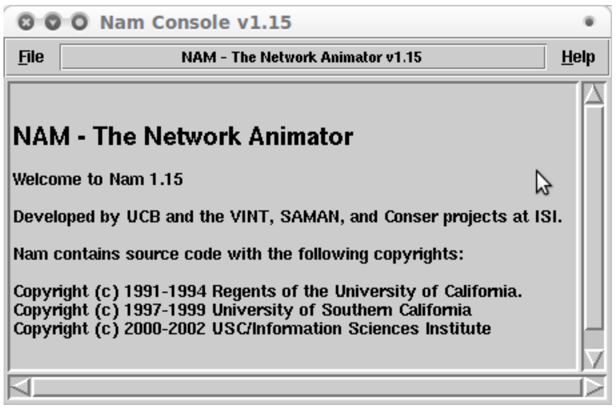
Compound TCP is a TCP variant protocol offering congestion control solution for high-speed and long distance networks. The key idea of CTCP is to add a scalable delay-based component to standard TCP. This delay-based component has a scalable window increasing rule that not only can efficiently use the link capacity, but can also react early to congestion by sensing the changes in RTT. If a bottleneck queue is sensed, the delay based component gracefully reduces the sending rate. This way, CTCP achieves good RTT fairness and TCP fairness.

Compound TCP(C-TCP) is widely deployed as it is the default transport layer protocol in the Windows operating system. The Compound protocol aims to use both queuing delay and packet loss as feedback to regulate its flow and congestion control algorithms. Compound maintains both cwnd (the loss window) and dwnd (the delay window). The loss window is the same as in the standard TCP Reno algorithm, which aims to control the loss based component. CTCP can efficiently use the network resource and achieve high link utilization. In theory, CTCP can be very fast to obtain free network bandwidth, by adopting a rapid increase rule in the delay-based component, e.g. multiplicative increase. CTCP has similar or even improved RTT fairness compared to regular TCP. This is due to the delay-based component employed in the CTCP congestion avoidance algorithm. It is known that delay-based flow, e.g. Vegas, has better RTT fairness than the standard TCP CTCP has good TCP-fairness. By employing the delay based component, CTCP can gracefully reduce the sending rate when the link is fully utilized. In this way, a CTCP flow will not cause more self-induced packet losses than a standard TCP flow, and therefore maintains fairness to other competing regular TCP flows.

III. Simulation and Emulation Basics

A. Simulation

Real-time simulation is a modeling technique where components (simulator objects) reproduce a timing behavior similar or equal to the timing behavior of the simulated targets (simulated entities). During the development of an application it interacts with the simulated environment in the same way it would interact with a real one. This allows testing it in different environments with relatively small effort before. The network simulator NS-2 is a widely accepted discrete event network simulator, actively used for wired and wireless network simulations. It Supports an array of popular network protocols and popularly used in the simulation of routing and multicast protocols, and is heavily used in ad-hoc research. It has an emulation feature, i.e. the ability to introduce the simulator into a live network using a soft real-time scheduler which tries to tie the event execution within the simulator with the real-time.



NAM - The Network Animator

Welcome to Nam 1.15

Developed by UCB and the VINT, SAMAN, and Conser projects at ISI.

Nam contains source code with the following copyrights:

Copyright (c) 1991-1994 Regents of the University of California.
Copyright (c) 1997-1999 University of Southern California
Copyright (c) 2000-2002 USC/Information Sciences Institute

Fig.2: Network Animator

NS2 is an open-source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks. It has many advantages that make it a useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic. Additionally, NS2 supports several algorithms in routing and queuing.

B. Emulation

Network emulation and simulation are widely used to develop, test, and debug new protocols, to explore and study a specific network-related research issue, or to evaluate the performance of an existing protocol or a scheme. Network emulation is the execution of real network protocol implementation code in a controllable and reproducible laboratory network environment. Unlike network simulation, the protocols and applications as well as the interaction between protocols are "real". Network traffic physically traverses the emulation environment, in which underlying protocols are tested and evaluated against user defined network conditions and traffic dynamics, such as packet latency, link bandwidth, packet drop rate, Bit Error Rate (BER), and link failure Network emulators are important tools for doing research and development related to network protocols and applications. With network emulation it is possible to perform tests of realistic network scenarios in a controlled manner, which is not possible by only using real network devices without emulation capabilities.

IV. Simulation Results

We have installed Network Simulator 2 (NS2) on Linux Ubuntu 12.04 LTS. We have run the TCL scripts of the different TCP variants Tahoe and Reno. Simulation results are as follows:

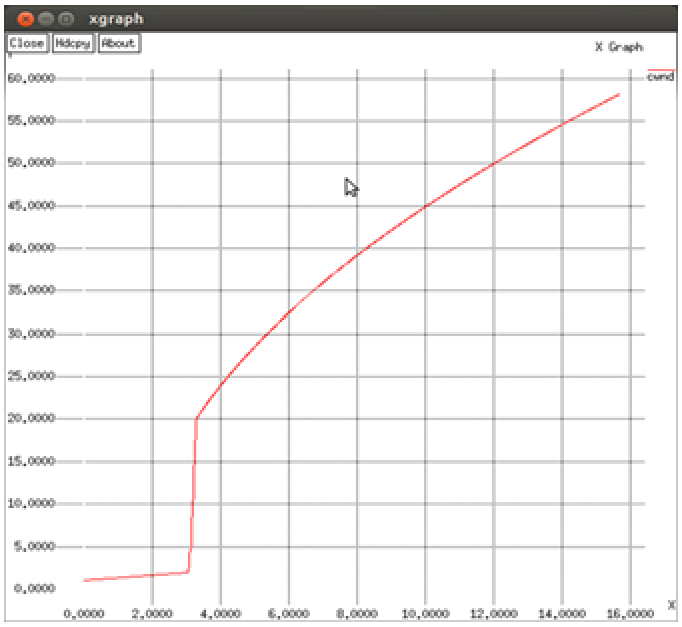


Fig.3: Tahoe TCP

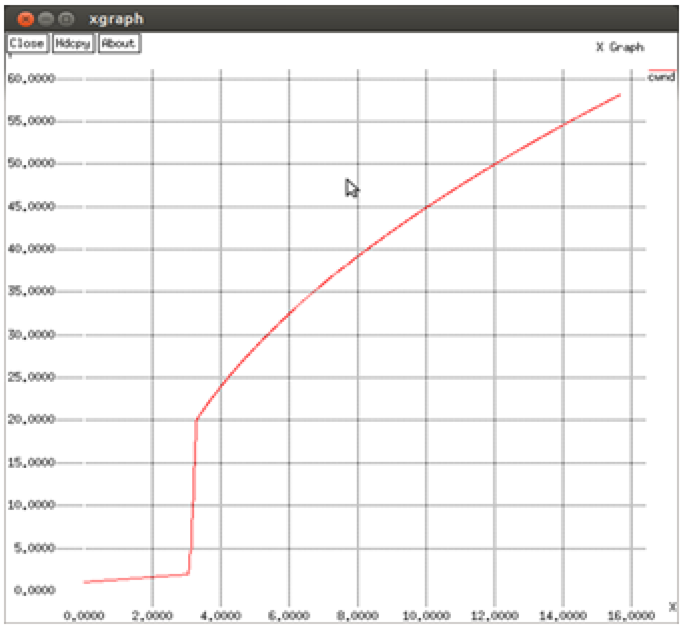


Fig.4: Reno TCP

V. CONCLUSION

TCP Tahoe does not deal well with multiple packet drops within a single window of data. TCP Reno introduced major improvements over Tahoe by changing the way in which it reacts to detecting a loss through duplicate acknowledgements. The idea is that the only way for a loss to be detected via a timeout and not via the receipt of a DUPACK is when the flow of packets and ACKs has completely stopped - This would be an indication of heavy congestion.

VI. FUTURE WORKS

We have compared the TCP TAHOE and RENO and concluded that the RENO has improved the performance. But Reno encounters several problems with multiple packet losses in a window of data (usually in the order of half a window). We could do a lot of work further in the research like working with readings obtained from other algorithm like New Reno, Sack, Westwood, Vegas, Compound, HS-tcp etc. This usually happens when invoking Fast Retransmit and Fast Recovery. This protocol comes as modification to the Reno but has not found implementation. We could get their readings and idea about the improvement factor with that protocol. We could modify the source code of the protocol itself in linux as it is basically a C program.

The modification in the window size and type and number of acknowledgement given will cause a radical change in the data rate, They were kept to give the best rates when made but with the development in the communication technology and services the modification can bring about a good change in the outcome. We could get the idea that which factor is the change in the specialized protocol like Westwood and Vegas and so on (as mostly they have only an algorithm that increases the throughput) and can look at the modification they have and we can cumulatively add this algorithm and can cause modifications in one protocol and cause a new protocol with much better throughput. Severe testing of each protocol can be done with Network Emulator software. Herein we have concentrated on the basic two network parameters delay and drop rate we can work with many different parameters with this emulation.

ACKNOWLEDGMENT

We would like to express our sincere gratitude and gratefulness to our professor Mr Kapil Raviya for providing us with all sorts of basic ideas and techniques essential for carrying out this project work from the very beginning to the end and enabled us to present this dissertation in this form. The teaching staffs also deserve our sincere thanks for sharing their discussion and exchange of ideas.

Lastly, we would like to extend our sincere gratitude to the known and unknown writers of the books and references that has been taken during the preparation of this project work.

VII. REFERENCES

[1] Hardik V. Miyani , Vishv B. Kukadiya, Mr. Kapil S. Raviya, Mr.Dhrumil Sheth , "Performance Based Comparision Of TCP Variants 'TAHOE, RENO, NEWRENO, SACK' In NS2 Using LINUX PLATFORM"

[2] Yuan-Chen Lai & Chan -Li Yao , "The Performance Comparison between TCP Reno and TCP Vegas"

[3] I Abdeljaouad, H. Rachidi, S. Fernandes, A. Karmouch , "Performance Analysis of Modern TCP Variants: A Comparison of Cubic, Compound and New Reno

[4] R. D. Mehta, Dr. C. H. Vithalani, Dr. N. N. Jani , "Evaluation Of TCP Variants – 'RENO' and 'SACK' On 'REAL TEST BED' Using EMULATOR – 'NISTNET'"

[5] Habibullah Jamal, Kiran Sultan, "Performance Analysis of TCP Congestion Control Algorithms"

[6] D. M. Lopez-Pacheco & C. Pham, "Performance comparison of TCP, HSTCP and XCP in high-speed, highly variable-bandwidth environments"

[7] Madhvi A. Bera, Bhumika S. Zalavadia, Rashmi Agrawal, "Effectuation of TCP Agents And Equivalence Of Outcome With Different TCP Variants"

[8] Mr Devang G. Chavda, Prof. Ridhdhi I. Satoniya, "Performance Evaluation of TCP in the Presence of UDP in Heterogeneous Networks by using Network Simulator 2"

[9] Haseen Rahman, Krishnamurthy Giridhar, Gaurav Raina, "Performance analysis of Compound TCP with AQM"

[10] Kun Tan Jingmin Song Qian Zhang, Murari Sridharan, "A Compound TCP Approach for High-speed and Long Distance Networks"

[11] Damien Phillips & Jiankun Hu, "ANALYTIC MODELS FOR HIGHSPEED TCP FAIRNESS ANALYSIS"

[12] http://www.icir.org/floyd/hstcp.htm

[13] http://www.isi.edu/nsnam/ns/

[14] http://www.isi.edu/nsnam/ns/ns-emulation.html

[15] http://www.linuxjournal.com/article/5929

Mr. Harshit B. Parmar has completed his Diploma Engineering in Electronics & Communication from Lukhdhirji Engineering College, Morbi, Gujarat, India. He is currently pursuing Bachelor of Engineering in Electronics & communication at C.U.Shah college of Engineering & Technology, Wadhwan, Gujarat, India. His area of research is in Digital Electronics, Embedded Systems & Networking.

Mr. Kapil S. Raviya has completed his bachelor and master's degree in the Electronics & Communication Engineering from Saurashtra University and Gujarat technological university Gujarat, India and he is currently pursuing his Ph.D. in the same discipline from C.U.Shah University, Wadhwan, Gujarat. His Area of research is image processing and computer vision system. He is currently working as an assistant professor in the department of electronics & communication engineering at C.U.Shah College of engineering and technology. He has presented 4 national and international papers and published 10 international papers. He has published a book titled Performance Evaluation of disparity map of stereo images with Lambart Publishing house, Germany. His area of interest is Image and video processing, Digital signal processing.