A Novel Scheme for Congestion Control in Hierarchical Mobile IPv6 Networks

P. Harini[†], B. Eswara Reddy[††], U.S.N. Raju[†††] and V. Vijaya Kumar[††††]

[†]Prof. and HOD, Dept. of IT, St. Anns College of Engg. & Technology, Chirala, A.P., India.
[††]Assoc. Prof. and HOD, Dept. of CSE, JNTUA College of Engineering, Anantapur, A.P., India.
[†††]Professor, Dept. of CSE, Rajeev Gandhi College of Engg. & Technology, Nandyal, A.P., India.
[††††]Professor, Dean, Dept. of CSE&IT, Head- SR Research Forum, GIET, Rajahmundry, A.P., India.

Summary
Packet losses occur repeatedly due to temporal link disconnection during handoff in wireless networks. Also during the handoff, the amount of bandwidth present at the new point of attachment may be different from that of the previous one. Due to this bandwidth change, packet drops or wastage of resources and congestion may occur at the new access point. To overcome this problem, the present paper proposes a TCP (Transport Control Protocol) based Path Loss Acknowledgment (TCP–PLACK) mechanism in place of TCP-SACK (selective acknowledgement) mechanism. The proposed TCP-PLACK mechanism sends a special acknowledgment which consists of packet loss details and available bandwidth at the new access point whenever a TCP receiver is attached to a new access point after a disconnection period or handoff. On receiving this acknowledgment, the sender retransmits the lost packets and adjusts the sending rate according to the bandwidth availability at the new access point.

Key words:
HMIPv6, Handoffs, Bandwidth, Throughput, Congestion.

1. Introduction

1.1 Mobile IP

In the early days of the Internet's development, a decision was made that Internet protocol (IP) addresses would represent both the topological location and identity of an end-host (RFC 791) (Pos81b). While this decision simplified the Internet's conceptual addressing model and met the needs of early network deployments, it has created difficulties for the development and deployment of truly mobile, IP-based applications and services. To reduce the handover delay and packet loss, many authors have suggested their ideas. Some concentrated on the link-layer [4] to detect the movement of Mobile Nodes (MN) as early as possible, others focus at network-layer [15] to accelerate the 'binding update process' by buffering and simulcasting packets.

S-MIP [10] provides a novel architecture that builds on top of the hierarchical approach and the fast handover mechanism, in connection with a newly developed handoff algorithm based on pure, 'software-based movement tracking techniques.' S-MIP introduces a new entity in the network, the Decision Engine (DE) that is similar to a MAP in its scope, and makes handover decision for its network domain. S-MIP provides improvement in both delay and packet loss, however, the operation of DE entity is difficult to simulate in test-bed and therefore the evaluation for this framework is not so far clear.

Shiao-li Tsao et al. [16] proposed a dynamic load balancing scheme for a voice over IP (VoIP) over WLAN (VoWLAN) system. The network-assisted association policy, by means of an Access point (AP), which instructs a station (STA) to request a VoWLAN session with the minimal load. The proposed load balancing scheme again rearranges the serving VoWLAN STAs between APs to allocate adequate resources for admitting the new request, if the APs are all overloaded. This rearranging mechanism is again a time consuming process.

Issam Jabri1 et al. [11] focused on the design of a QoS management solution for wireless communication systems. For better performances of the Wireless LAN, this method needs a balanced distribution of mobile stations among the available Access Points. The main disadvantage of this approach is estimation of balanced distribution.

Antonios Argyriou [4] presents a joint performance evaluation model of TCP and TFRC, with the fundamental IP-based mobility protocols. The protocol performance during handoffs between heterogeneous wireless networks like WLAN, cellular, or WMAN are characterized by the stochastic models developed. This method is too complex in nature.

Ezil sam Leni A. et al. [5] proposed a new technique based on the information from the Data Link Layer and Network Layer. This reduces the handoff delay and increases the TCP performance in integrated wireless networks. To identify the Neighbour Routers during handover, the

Manuscript received February 5, 2011
Manuscript revised February 20, 2011

Candidate Access Route Discovery (CARD) Protocol is used.
Andrei Gurtov et al. [3] evaluated performance of TFRC during handovers between GPRS, WLAN, and UMTS.
The above standard and non-standard solutions are proposed for making the handover seamless. The solutions proposed by various authors obviously improved the handover performance to a little or some extent, especially the latency. But in practice the handover delay is still very high for time-sensitive services. The main purpose of the proposed paper is to analyze existing handover implementations and to propose suitable improvements to the Hierarchical Mobile IPv6 protocol architecture so that latency of the handover is minimized.
The present paper is organized as follows. Section 2 deals with the basic concepts of congestion control in HMIPv6, section 3 introduces the proposed new TCP-PLACK mechanism, section 4 deals with results and discussions and section 5 deals with conclusions.

2. Congestion Control

The TCP congestion control mechanism has been continuously developed for over 15 years. There are a large number of standardized and non-standardized TCP variants nowadays; however, in this section, only the most frequently used standardized TCP congestion control algorithms in the current Internet are described, to help the understanding of the impact of mobility on TCP performance. The current basic TCP congestion control mechanism consists of four algorithms as specified in [2]: slow start, congestion avoidance, fast retransmit and Reno-fast recovery. These algorithms suffer with many disadvantages that affect the TCP performance in dealing with multiple segments. For example, if the ACK that acknowledges new data does not acknowledge all the segments before the fast retransmit is invoked, these algorithms cannot recover efficiently since the fast retransmit algorithm only leads to the retransmission of one segment [2] [7]. To overcome this NewReno fast recovery algorithm is proposed.

2.1 New Reno TCP Fast Recovery Algorithm

During the Reno fast recovery procedure, however, if the ACK that acknowledges new data does not acknowledge all the segments before the fast retransmit is invoked, the Reno algorithm cannot recover efficiently since the fast retransmit algorithm only leads to the retransmission of one segment [2] [7]. Therefore, the NewReno fast recovery algorithm [7] was proposed to improve TCP performance when multiple segments are lost from a window of data, in the absence of explicit information on which segment(s) should be retransmitted, such as the information provided by the TCP Selective Acknowledgement (SACK) options [13] when they are supported. The NewReno fast recovery algorithm introduces the concepts of partial acknowledgement and full acknowledgement during the fast recovery procedure. A partial acknowledgement is referred to as the ACK that acknowledges a retransmitted segment but not all of the segments transmitted before the fast retransmit, while a full acknowledgement is referred to as the ACK that acknowledges not only a retransmitted segment, but also all of the segments transmitted before the fast retransmit. The key idea of the NewReno algorithm (also the key difference between the NewReno and the Reno algorithms) is to distinguish the response to a full acknowledgement and a partial acknowledgement. On receipt of a full acknowledgement, *cwnd* is deflated to either the minimum of *ssthresh* and *FlightSize+SMSS*, or just *ssthresh* as the Reno algorithm specifies. The choice is implementation dependent. Then the fast recovery pocedure ends. On receipt of a partial acknowledgement, the TCP sender immediately retransmits the first unacknowledged segment. If this is the first partial acknowledgement during the fast recovery procedure, the TCP sender also resets the retransmit timer. This is called the impatient variant of NewReno. If the retransmit timer is reset on every partial acknowledgement, it is called the slow-but-steady NewReno. In the impatient variant of NewReno, if the fast recovery procedure does not end before the retransmit timer reaches the value of RTO, a slow start procedure is invoked. [7] At the same time, *cwnd* is deflated by the amount of cumulative acknowledged new data. If the partial acknowledgement acknowledges at least SMSS of new data, *cwnd* is then artificially inflated back by SMSS to reflect the number of outstanding segments being reduced by one. A new segment is transmitted at this time if the minimum of *cwnd* and *rwnd* allows. These operations repeat until a full acknowledgement arrives or the retransmit timer reaches the value of RTO. In this way, the NewReno algorithm improves TCP performance in the case when multiple segments are lost from one window of data. The NewReno algorithm has been widely deployed in the current Internet. Tests have been done to prove that the NewReno algorithm is used more than the Reno algorithm in SACK-incapable TCP implementations [7].

2.2 Selective Acknowledgement (SACK) options for TCP

Although the NewReno algorithm can improve the Reno algorithm performance in the case when multiple segments are lost from one window of data, it does still have its own disadvantages. For example, in the situation that no segments are lost but just reordered, the NewReno

algorithm can result in more unnecessary retransmissions than the Reno algorithm. This is because the NewReno algorithm is based on limited information obtained from ACKs and conjectures on which segments may be lost. It is only deemed as a good choice when the TCP SACK mechanism is not available [7]. The TCP SACK mechanism [13] allows the TCP receiver to include a SACK option in an ACK packet, explicitly informing the TCP sender of which out-of-order segments have been received. Then the TCP sender can only retransmit those missing segments. One SACK TCP implementation that requires minimum changes to the Reno algorithm is described in [6]. Its main difference from the Reno algorithm lies in the behavior when multiple packets are lost from one window of data. In general, this SACK TCP implementation (simply called the SACK algorithm hereafter) enters the fast retransmit procedure when a sufficient number (normally three) of duplicate ACKs are received, as in the Reno algorithm; and distinguishes the response to a full acknowledgement and a partial acknowledgement, as in the NewReno algorithm. In the case that a retransmitted packet is dropped, the SACK algorithm waits for an RTO. A slow start procedure is then initiated and the dropped packet is retransmitted again. When a full acknowledgement arrives at the TCP sender, the SACK fast recovery procedure ends. Fall . K. etal compares the Reno, NewReno and SACK algorithms based on extensive simulations, and the results show that the SACK algorithm has the best overall performance in various scenarios.

Although TCP-SACK improves the performance, it does still have disadvantages, especially when temporal link disconnects. This results packet drops and ultimately leads to congestion. To overcome these drawbacks of TCP-SACK mechanism the present paper proposes a new TCP-PLACK mechanism.

3. New TCP-PLACK mechanism

To accommodate TCP to more real handoff situations, the present paper proposes a path loss acknowledgment (TCP–PLACK) mechanism. In TCP-PLACK mechanism, whenever a TCP receiver is attached to a new access point after a disconnection period or handoff, it sends this special ack which consists of three components:

(i) A PLFLAG, and

(ii) A TCP SACK option [13]

(iii) A bandwidth availability field (ABW)

The information about sequence numbers of the lost packets is stored in TCP SACK. TCP SACK then informs the TCP sender about the details of packets lost due to this temporal link disconnection. Bandwidth availability field is used to provide an efficient estimation of the bandwidth availability in the new network. Because of this, within one round trip time (RTT), senders can adjust their sending rates to the most suitable value. Thus the rate adjustment is done either by increasing their transmission rates to make full utilization of the new network resources or by decreasing their transmission rates to avoid overloading the new network with bursty traffic.

3.1 The Path Loss Acknowledgement (PLACK) Mechanism

The basic objective of TCP-PLACK is to recover the harmful impact of the temporal link disconnection due to a handoff and to balance the network bandwidth. A TCP receiver on MN notifies its TCP sender on a FN (fixed node) of the temporal disconnection through a special acknowledgement, to reduce the influence of packet losses during the disconnection, then, the TCP sender retransmits the lost packets and restores the reduced Congestion Window (cwnd) and sends Slow Start Threshold (ssthresh) signal if a false Retransmission Time Out (RTO) occurs.

TCP–PLACK requires modifying both the sender's and receiver's operations on the TCP–SACK option. Some fields included in TCP–PLACK are listed as follows:

- PLFLAG: This flag plays a role in notifying the sender that a wireless link was temporally disconnected or reconnected. The flag is set to one of the three values: no loss (2), partial packet loss (1), and full packet loss (0).
- Ccwnd: A copy of cwnd is used to restore the cwnd if an RTO occurred due to a disconnection.
- Cssthresh: A copy of ssthresh is used to restore ssthresh if an RTO occurred due to a disconnection.
- Stime: Indicates the last packet transmitted from the TCP sender within a window due to a RTO or a fast retransmits.
- PLtimer: This timer is maintained in order to notify the TCP sender about the loss of all packets.
- ABW: The bandwidth availability (ABW) field is used to provide an efficient estimation of the bandwidth availability in the new network.

In the proposed TCP-PLACK mechanism, the receiver reacts to a temporal disconnection as follows:

With the help of the link layer and Mobile IP, the recovery of the wireless link is identified by the receiver. The sender is informed by the receiver about the reconnection of the wireless link by sending PLACK with PLFLAG set to 2. The sender immediately checks if the sender's retransmission timer expires or if one RTT has elapsed after the Stime. Then, the receiver schedules the PLtimer

with RTT/2 since any additional packets are anticipated to be delivered within RTT/2, if they exist. If a new packet arrives at the receiver before the PLtimer expires it includes gaps in the sequence numbers. To transmit only the lost packets by the sender, the receiver sends an acknowledgement packet with a PLFLAG set to 1 and a SACK block. Then, the receiver cancels the PLtimer. When the PLtimer expires and no new packets have arrived at the receiver, the receiver sends an acknowledgement packet with only the PLFLAG set to 0. This insists the sender to send all unacknowledged packets or to perform the slow start algorithm. Based on this Receiver calculates the available bandwidth of the new AP in the following way:

Let W be the capacity of the wireless network. Idle rate (IR) indicates the rate at which the link is idle. Then the available bandwidth (ABW) can be obtained by the following product:

$$ABW=W\times IR \quad (1)$$

The busy time of the link can be estimated by adding up all the transactions of nodes in the network as given by the equation (2).

The transaction time of node i can be obtained via the sum of the sending and receiving times to/from node i ($T_s(i)+T_r(i)$) where $T_s(i)$ is the sending time from node i to j and $T_r(i)$ is the receiving time from node j to i. Transaction time between other nodes can be obtained on looking time $T_o(i)$, from the NAV in node i that is updated in other node transactions. The present paper estimated the busy time (BT) of any link i in the network and given in Equation (2):

$$BT_l=T_s(i)+T_r(i)+T_o(i) \quad (2)$$

The idle rate using the busy time is given by the equation (3).

$$IR=1-(BT_l/TT) \quad (3)$$

where TT is the total elapsed time.

Based on Equation (1), (2) and (3) the ABW is given in Equation (4).

$$ABW=W\times 1-(BT_l/TT) \quad (4)$$

3.2 The Sender's Operations

When the lost packets are recovered by the sender it begins to receive an acknowledgement with a PLACK. The detailed operations of the sender are given by the following three steps.

Step 1: The sender checks the value of the PLFLAG when a retransmission timer is pending and the sender receives a PLFLAG, and does the following actions as specified in steps 1.1, 1.2, 1.3, 1.4 and 1.5.

Step 1.1: If the PLFLAG is set to 2, then (Stime+RTT) the current time is compared by the sender. If current time is larger than (Stime+RTT), then TCP sender considers as all packets are lost and then retransmits all unacknowledged packets. Otherwise, the sender does nothing. If the sender transmitted the lost packets, the process moves to Step 1.4.

Step 1.2: If the PLFLAG option is set to 0, the sender follows the operations of Step 1.1. However, if current time is smaller than (Stime+RTT), then the sender cancels the pending retransmission time and enters the slow start algorithm.

Step 1.3: If the PLFLAG is set to 1 and SACK blocks are included, the sender gets back the sequence numbers of the lost packets from each SACK block. Then it retransmits the lost packets to the receiver immediately. In order to prevent the timer expiry during the recovery of the lost packets, it resets the retransmit timer. If the sender transmitted the lost packets, the process moves to Step 1.4.

Step 1.4: After transmitting the lost packets, when the sender receives duplicate ACKs, it transmits a new packet for each dup ack if congestion window is available.

Step 1.5: If the retransmitted packets by the first PLFLAG are still traversing and if the retransmission timer is pending, then all other PLFLAG values are ignored. The sender continues to send new packets when an incoming acknowledgement is arrived, if the retransmitted packets successfully arrive and the sender receives a new non-duplicate acknowledgement from the receiver.

Step 2: When the RTO has already occurred and sender receives a PLFLAG, the sender replaces cwnd and ssthresh with Ccwnd and Cssthresh, respectively. With an enlarged cwnd, the sender continues the normal operation of TCP by flushing both the unacknowledged packets and new packets into the network.

Step 3: If the available bandwidth of the new network ABW_{new} is greater than that of the old one ABW_{old}, the congestion window increases by (1/cwnd) for each PLACK received, i.e., (cwnd=cwnd+(1/cwnd)) until a congestion occurs. On the other hand, if ($ABW_{new}<ABW_{old}$), then the new network suddenly gets overloaded with a large number of data packets. This congests, in turn, the transmission queue at the bottleneck link's router and eventually results in the discard of a large number of packets. As a result, TCP almost immediately decreases its cwnd to 1.

4. Experimental Results

4.1 Simulation Setup

This section evaluates the performance of TCP–PLACK using the ns-2 simulator. The TCP–SACK1 agent and its sink agent in ns-2 are modified to implement TCP–PLACK. The receiver's window size is set to 100 packets in order to receive extra packets during the lost packet recovery. The link layer connector and mobile host agent in ns-2 are modified to notify the TCP sink of the wireless link reconnection. The queue size of each node is set to 100 packets. The entire simulation settings are given in Table 1. The simulation was carried out using the hierarchical mobile network topology shown in Fig. 2.

Table1: Simulation Settings.

No. of Nodes	15
Area Size	1000 X 1000
Mac	802.11
Simulation Time	50 sec
Traffic Source	FTP
Packet Size	100,200,….500 bytes
Speed	20
Transmission range	75m
Routing Protocol	AODV

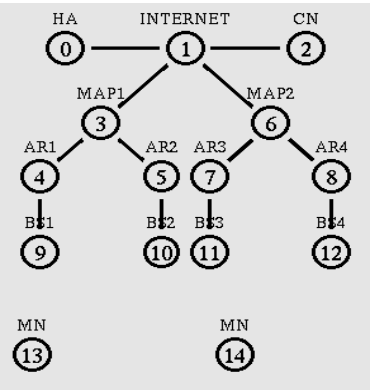

Fig. 2 Network Topology.

Initially the mobile node MN13 was in MAP1 in the domain AR1. During the simulation we perform intra and inter domain handoff on MN13. Initially, at time t1, the mobile node performs intra domain handoff by moving from AR1 to AR2 within MAP1. Next at time t2, it starts moving towards AR3 from AR2, thus by performing inter domain handoff. Here AR3 is in MAP2, which has lower bandwidth than MAP1. Hence the new network will become congested leading to packet drops and throughput degradation. At time t3, it moves from AR3 to AR4, within MAP2. Finally at time t4, it moves back to AR1, once again performing inter domain handoff.

In our first experiment, the simulations are performed by varying the disconnection period from 0.25s to 1.25s. The first disconnection begins at 2seconds (2s) after the initiation of the simulation.

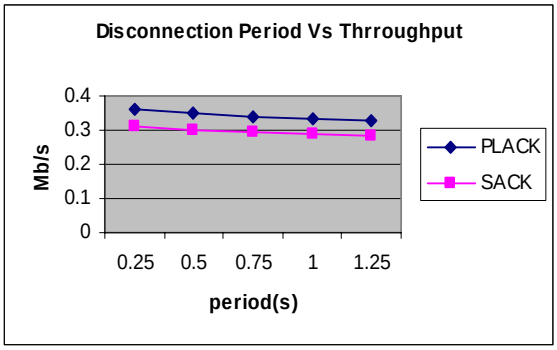

Fig. 3 Disconnection Period Vs Throughput.

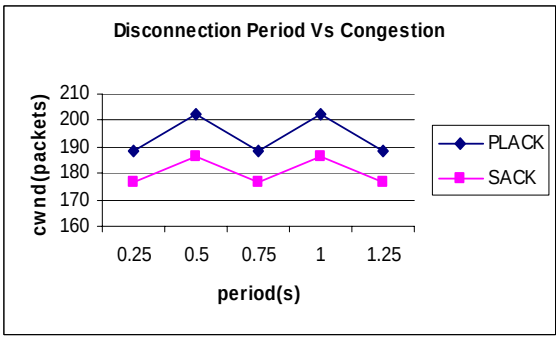

Fig. 4 Disconnection Period Vs Congestion.

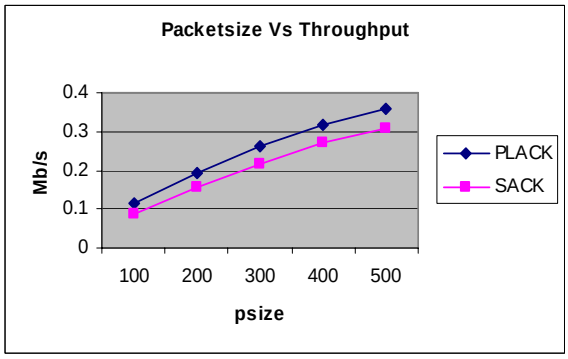

Fig. 5 Packet size Vs Throughput.

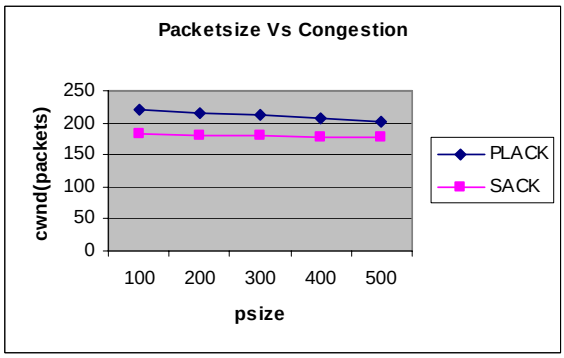

Fig. 6 Packet size Vs Congestion.

Fig. 3 shows the average throughput obtained by the receiver during the entire simulation. It clearly shows that the proposed PLACK mechanism gains more throughput compared to the SACK when the disconnection period is increased. Fig. 4 shows the average congestion window size in terms of packets. From the Fig. 4, one can observe that PLACK accumulates more packets in its congestion window, when compared to SACK. In our second experiment, the simulations are performed by varying the TCP packet size from 100 bytes to 500 bytes. Fig. 5 shows the average throughput obtained by the receiver during the entire simulation. It clearly shows that PLACK gains more throughput compared to the SACK when the packet size is increased. Fig. 6 shows the average congestion window size in terms of packets. From the Fig. 6 it is clearlyevident that PLACK accumulates more packets in its congestion window, when compared to SACK.

5. Conclusions

This paper proposed a novel Cross-Layer based scheme for loss recovery and rate control for handoff in hierarchical Mobile IPv6 networks. In this scheme a path loss acknowledgment (TCP–PLACK) mechanism is proposed to accommodate TCP to more real handoff situations. When a TCP receiver is attached to a new access point after a disconnection period or handoff, it sends a special acknowledgment which consists of three components, a PLFLAG, a TCP SACK and a bandwidth availability field (ABW). By the introduction of these three fields in TCP-PLACK within one round trip time (RTT), senders can adjust their sending rates to the most suitable value. Thus the rate adjustment is done either by increasing their transmission rates to make full utilization of the new network resources or by decreasing their transmission rates to avoid overloading the new network with bursty traffic. The simulation results are carried out for the proposed TCP-PLACK and TCP-SACK mechanism based on disconnection period vs throughput, disconnection period vs congestion, packet size vs congestion. The simulation results shows the efficacy of the proposed method over TCP-SACK.

Acknowledgments

The authors would like to express their cordial thanks to K.V.V. Satya Narayana Raju, Chairman, Chaitanya Institutions and K. Sashi Kiran Varma, Secretary, GIET, Rajahmundry for providing Research facilities. Authors would like to thank Dr. G.V.S. Anantha Lakshmi for her invaluable suggestions and constant encouragement that led to improvise the presentation quality of the paper.

References

[1] "Mobile IPv6 Overview", Cisco Systems, Dec 2004.
[2] Allman M., Paxson V., and Stevens W., "TCP Congestion Control", RFC 2581, April 1999.
[3] Andrei Gurtov and Jouni Korhonen, "Effect of Vertical Handovers on Performance of TCP-Friendly Rate Control," ACM SIGMOBILE Mobile Computing and Communications, Volume 8, Issue 3, pp: 73 – 87, July 2004.
[4] Antonios Argyriou, "A joint performance model of TCP and TFRC with mobility management protocols," Wireless Communications & Mobile Computing Volume 6 , Issue 5, pp: 547 – 557, August 2006.
[5] Ezil sam Leni A. and Srivatsa S.K., "A handoff technique to improve TCP performance in next generation wireless networks," Inform. Technololgy Journal, vol. 7, pp: 504-509, 2008.
[6] Fall K. and Floyd S., "Simulation-based Comparison of Tahoe, Reno, and SACK TCP", Computer Communication Review, July 1996.
[7] Floyd S., Henderson T., and Gurtov A., "The New Reno Modification to TCP's Fast Recovery Algorithm", RFC 3782, April 2004.
[8] Goff T., Moronski J., Phatak D., Gupta V., "Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments", INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, vol. 3, 1537 – 1545, Mar 2000.
[9] Hesham Soliman, Claude Castelluccia, Karim El- Malki, Ludovic Bellier, "Hierarchical Mobile IPv6 mobility management (HMIPv6)," draft-ietf-mipshophmipv6- 00.txt. IETF Mobile IP Working Group, June 2003.
[10] Hsieh R., Zhou Z.G., and Seneviratne A., "S-MIP: A seamless Handoff Architecture for Mobile IP," IEEE INFOCOM, 2003, pages 1774 -1784, volume 3.
[11] Issam jabri1, thierry divoux, nicolas krommenacker, adel soudani, "IEEE 802.11 Load balancing: an approach for QoS Enhancement," International Journal of Wireless Information Networks, 2008.
[12] Johnson D., Perkins C., Arkko J., "Mobility Support in IPv6," Draft-ietf-mobileip-ipv6-24.txt, IETF Mobile IP Working Group, Dec.2003.
[13] Mathis M., Mahdavi J., Floyd S., Romanow A., "TCP Selective Acknowledgement Options," RFC, 2008.
[14] Narten T., "Neighbor Discovery for IP Version 6 (IPv6)," RFC 2461, IETF, 1998.
[15] Olivia Brickley, Susan Rea, Dirk Pesch, "Load Balancing for QoS Enhancement in IEEE802.11E WLANs Using Cell Breathing Techniques," 7th IFIP International Conference on Mobile and Wireless Communications Networks, Maroc, 2005.
[16] Shiao li tsao and chih chien hsu, "A Dynamic Load Balancing Scheme for VoIP over WLANs," journal of information science and engineering, vol. 24, pp: 47-60, 2008.
[17] Soliman H. et al., "Hierarchical MIPv6 mobility management (work in progress)," IETF Internet-Draft, draft-ietf-mobileip-hmipv6-07.
[18] Thomson S., "IPv6 Stateless Address Auto configuration," RFC 2462, IETF, 1998.

P. Harini M.Tech (Remote Sensing), M.Tech. (CSE), [Ph.D. (Mobile Computing)]. I obtained my M.Tech. (Remote Sensing) in 1997 & M.Tech. (CSE) in 2003 from JNTU, Masab Tank, Hyderabad. I worked as a Research Associate in JNTU, Masab Tank, Hyderabad in Remote Sensing Department for 01 year, 05 years worked as a Assistant Professor in QIS College of Engineering, Ongole and 01 year worked as a Associate Professor in Rao & Naidu Engineering College, Ongole. At present I am working as Professor & Head of the Computer Science and Engineering Department in St. Ann's College of Engineering & Technology, Chirala.

B. Eswara Reddy received the B.Tech. (CSE) degree from Sri Krishna Devaraya University in 1995, M.Tech. (Software Engineering) from JNT University in 1999 and Ph.D degree from JNT University in Computer Science. He worked as a Lecturer for two years (1996-97). He worked as Assistant Professor (from 1999) in the Dept. of Computer Science & Engineering, JNT University College of Engineering. At present he is working as Associate Professor and Head, in CSE Dept at JNTUCE, Anantapur (Since 2006). His research interests includes Image processing, Pattern Recognition and Software Engineering. He is a life member of ISTE and IE.

U.S.N Raju received the B.E. (CSE) degree from Bangalore University in 1998. He worked as a Software Engineer in INDIGO RDBMS Research and Development for two years (1999-2000). After that he completed his M. Tech. (Software Engineering) from JNT University in 2002. He worked as an Academic Assistant in JNT University, Hyderabad for six months and joined as an Assistant Professor in Mahatma Gandhi Institute of Technology, Hyderabad and worked there for five years (2002-2007). He worked as an Associate Professor in Godavari Institute of Engineering and Technology, Rajahmundry for three years. He completed his Ph.D. in Computer Science and Engineering from JNT University Kakinada under the guidance of Dr V. Vijaya Kumar in Feb'2010. Presently, he is working as a Professor at Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, Andhra Pradesh, India. He is a life member of ISTE, ISCA and CSI.

Vakulabharanam Vijaya Kumar received integrated M.S. Engg, degree from Tashkent Polytechnic Institute (USSR) in 1989. He received his Ph.D. degree in Computer Science from Jawaharlal Nehru Technological University (JNTU) in 1998. He has served the JNT University for 13 years as Assistant Professor and Associate Professor and taught courses for M.Tech students. He has been Dean for Dept of CSE and IT at Godavari Institute of Engineering and Technology since April, 2007.His research interests include Image Processing, Pattern Recognition, Network Security, Steganography, Digital Watermarking, and Image retrieval. He is a life member for CSI, ISTE, IE, IRS, ACS and CS. He has published more than 120 research publications in various National, Inter National conferences, proceedings and Journals.