ATCP: Improving TCP performance over mobile wireless environments

Ajay Kr Singh
Dept of Computer Science & Engg
IIT Bombay
aksingh@cse.iitb.ac.in

Sridhar Iyer
School of Information Technology
IIT Bombay
sri@it.iitb.ac.in

***Abstract*— Transmission Control Protocol (TCP) is known to suffer from performance degradation in mobile wireless environments. This is because such environments are prone to packet losses due to high bit error rates and mobility induced disconnections. TCP interprets packet losses as an indication of congestion and inappropriately invokes congestion control mechanisms, leading to degraded performance.**

While there are several proposals to optimize TCP in the presence of high bit error rates and mobility, they focus mainly on scenarios where the TCP sender is a fixed host. In this paper we propose ATCP, an approach which mitigates the of degrading effect of host mobility on TCP performance for *two-way data transfers*, i.e. scenarios where the TCP sender is a mobile host, in addition to scenarios where the TCP sender is a fixed host.

ATCP uses network layer feedback in terms of *disconnection* and *connection* signals, to modify the congestion control mechanisms of TCP, thereby achieving enhanced throughput in mobile wireless environments. We have compared ATCP with 3-dupacks (3DA) [10], Freeze TCP [4] and TCP Reno, by simulations using ns-2. We show that ATCP achieves an improvement of up to 40% over TCP Reno in WLAN environments and up to 150% in WWAN environments in both directions of data transfer.

***Index Terms*—TCP adaptation, Mobility, Wireless Networks, Network feedback, Network simulation, Performance study.**

I. Introduction

Transmission Control Protocol (TCP) [11] is a reliable, connection-oriented, full-duplex, transport protocol widely used in wired networks. TCP's flow and congestion control mechanisms are based upon the assumption that packet loss is an indication of congestion. While this assumption holds in wired networks, it does not hold in the case of mobile wireless networks. As shown in figure 1, a typical wireless mobile network has mobile hosts (MH) connected to base stations (BS) over wireless links. The base stations are inter-connected via wired network. Other fixed hosts (FH) may also be part of the wired network. In such networks, packet losses also occur due to *bad wireless channel conditions* and *intermittent disconnection introduced by mobility of the MH*. TCP interprets these packet losses as congestion and invokes the congestion control mechanisms, which reduce the *sending window* multiplicatively (by half), thereby reducing the sending rate drastically. However, *bad channel conditions* and *intermittent disconnections* are typically transient phenomena. Hence the TCP congestion control response is inappropriate and also undesirable as it decreases the throughput, resulting in an under-utilization of the network. While several approaches have been proposed for mitigating the effect of adverse channel conditions [1] [3] [5] [8] [9], there are few approaches [4] [6] [10], which tackle mobility induced disconnection. However, most of these schemes focus on scenarios where the TCP sender is a FH and do not perform well when the sender is a MH. Also, some of these approaches are vulnerable to scalability issues as they require per-flow support from the base station [9], or require modification to TCP at the FH [3]. In this paper we present **ATCP**, an adaptation to TCP, which alleviates the degrading effect of mobility on TCP performance. ATCP is designed for improving TCP performance in *two-way data transfers*, i.e., from MH to FH as well as from FH to MH. ATCP requires modifications to TCP at the MH only, and uses feedback from network layer about the mobility status in terms of *connection event* and *disconnection event* signals, to modify the TCP congestion control mechanisms appropriately. ATCP uses these signals to appropriately freeze/continue ongoing data transfer and changes the action taken at RTO (*Retransmission TimeOut*) event, leading to enhanced TCP throughput. We have compared ATCP with 3-dupacks (3DA) [10], Freeze TCP [4] and TCP Reno, by simulations using ns-2. We show that ATCP achieves an improvement of up to 40% over TCP Reno in WLAN environments and up to 150% in WWAN environments in both directions of data transfer. This paper is organized as follows: Section II, discusses some related work and motivates the need for our approach. Section III describes our approach ATCP, Section IV presents the simulations results and Section V gives the conclusions.

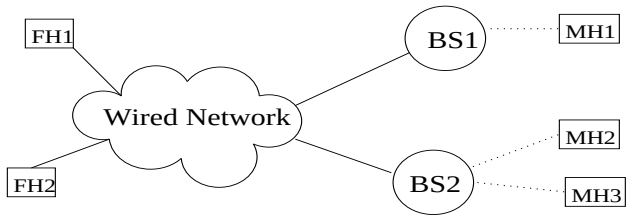

Fig. 1. A typical mobile wireless network

II. Related work

There are several approaches for mitigating the effect of adverse channel conditions on TCP performance [1] [3] [5] [8] [9]. We focus on approaches that attempt to reduce the detrimental effect of host mobility on TCP perfor-

mance [4] [6] [10]. Of these, we are primarily interested in approaches that require modifications only at the mobile host [4] [10]. Our approach ATCP falls into this category.

The 3-dupacks approach (3DA) [10] requires the network layer to provide information about ongoing mobility to the TCP layer at the mobile host (MH). After disconnection and upon subsequent reconnection, the MH sends three duplicate acknowledgements (dupacks) to the fixed host (FH). These dupacks cause the TCP sender at the FH to immediately enter the *fast recovery phase*, instead of waiting for its retransmission timer to expire. Thus this approach reduces the "idle period" of the TCP sender after the connection is re-established. However, the TCP sender at FH also reduces its slow start threshold (*ssthresh*) and congestion window (*cwnd*) parameters when it enters fast recovery phase. This side effect in turn reduces the throughput of the connection.

The Freeze TCP approach [4] requires the network layer at the MH to give an indication of impending disconnection. Upon receipt of such an indication, Freeze TCP at the MH sends a *zero window advertisement* to the FH. Upon reconnection, it uses 3DA [10] to restart transmission. The main drawback is that Freeze TCP requires the network layer to *predict* future disconnections. There is also the issue of how early should this prediction be made available to TCP at the MH. If it is available earlier than the RTT of the connection, Freeze TCP's action may lead to degraded performance. Since different connections may have different RTT values, this adds to the difficulty in accurate predictions.

In addition to 3DA and Freeze TCP, we have also compared the performance of ATCP with TCP Reno. Since TCP Reno is quite well-known and is described in standard text books [11], we omit discussing it here.

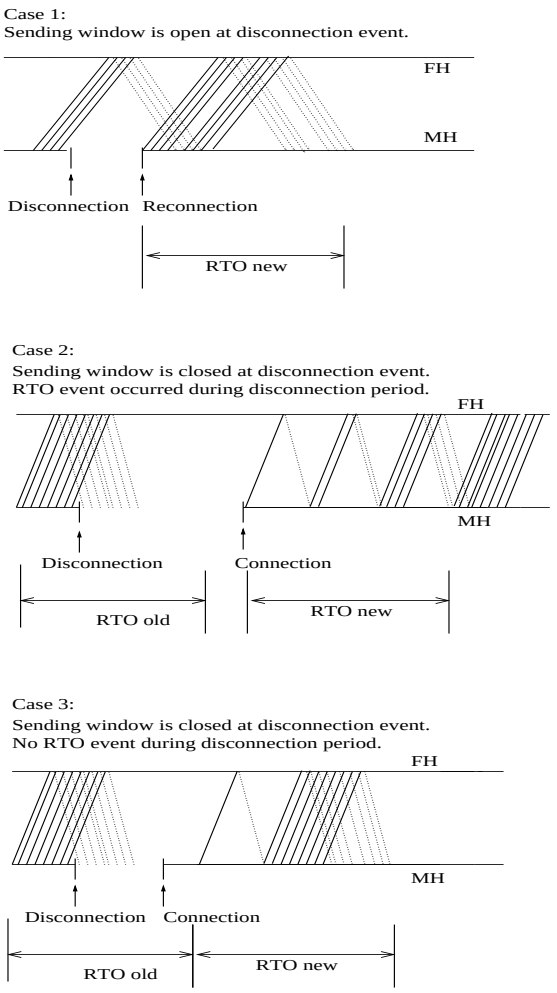

Fig. 2. ATCP behaviour under different disconnection scenarios

III. ATCP MECHANISM

ATCP is designed to improve TCP performance in wireless mobile networks in the presence of temporary disconnections caused by mobility. Unlike earlier work, ATCP improves the performance not only when the TCP sender is a fixed host (FH), but also when the TCP sender is a mobile host (MH). ATCP involves modifications to the network stack only at the MH and requires network layer feedback regarding the status of the connectivity.

ATCP assumes that the network layer sends a *connection event* signal to TCP when MH gets connected to the network and a *disconnection event* signal when the MH gets disconnected from the network. We believe that this is a reasonable assumption since such information is typically available with the network layer in wireless mobile networks (for instance, Mobile IP [7]). ATCP uses these signals to appropriately freeze/continue ongoing data transfer and changes the action taken at RTO (*Retransmission TimeOut event*), leading to enhanced TCP throughput.

In brief, the working of ATCP is as follows:

- MH to FH data transfer:
  - Upon *disconnection event*, if the sending window is open (case 1, figure 2), ATCP does not wait for ACK for packets sent before disconnection, and cancels the retransmission timer (RTX). If the sending window is closed and it was waiting for ACKs, ATCP does not cancel RTX but waits for the occurrence of an RTO event.
  - Upon *connection event*, if the sending window is open, ATCP sends data and sets a new RTX. Since ACKs are cumulative, the ACK for the new data also acknowledges the data sent before disconnection. If the sending window is closed and RTO has occurred, ATCP retransmits. If RTO has not occurred, ATCP waits for an RTO event.
  - Upon *RTO event*, ATCP checks whether a disconnection has occurred. If so (case 2, figure 2), ATCP instead of reducing *ssthresh* (typical TCP behavior), sets *ssthresh* to the value of *cwnd* at the time of disconnection and sets *cwnd* to one. If MH is connected to the network (case 3, see figure 2), ATCP retransmits the lost packet without modifying *ssthresh* or *cwnd* parameters.

  These actions enable ATCP to quickly regain the *cwnd* value prior to disconnection, thus reducing under utilization of the available link capacity.
- FH to MH data transfer: ATCP delays the ACK for the last two bytes by **d** milliseconds, (at most 500 milliseconds [12]).
  - Upon *disconnection event*, the network connectivity status is updated.

– Upon *connection event*, ATCP ACKs the first of these bytes with zero window advertisement (ZWA) and ACKs the second byte with a full window advertisement (FWA). *TCP at FH will process these ACKs as they have a higher sequence number* [12] than all previous ACKs. ZWA causes TCP sender at FH to freeze its RTX, without reducing its *cwnd*. FWA results in the TCP sender at FH retransmitting all unacknowledged packets with no reduction in *cwnd*.

These action prevent the TCP at FH from taking congestion control measures when packets are lost due to MH being disconnected.

The ATCP algorithm is presented below.

Events and Initialization for ATCP

Events

SendData: Application layer delivers data to TCP for transmission.
DataRcvd: Data packet received from peer TCP layer.
AckRx: TCP has received ACK from peer TCP layer.
RTO: Retransmission Timer has expired.
Disconnection: TCP layer receives disconnection signal from lower layer.
Connection: TCP layer receives connection signal from lower layer.

Initializations

Network Status = `connected`
StatusRecorded = `idle`
DisconnectionOccurred = `FALSE`
RTOoccurred = `FALSE`
d = `200ms`
AckPending = *FALSE*

Function calls

GetStatus()

if packets in send buffer & sending window is open **then**
  Return sending ;
end if
if sending window is closed & waiting for ACKs **then**
  Return waiting;
end if
Return idle ; /* *connection is idle* */

ATCP: MH to FH data transfer

SendData()
if $NetworkStatus == connected$ **then**
  Normal TCP behaviour;
else
  Buffer the data;
end if

Disconnection()
NetworkStatus = Disconnected;
StatusRecorded = GetStatus(); /* *record whether sender was Idle, Sending, Waiting* */
if $StatusRecorded == Idle$ **then**
  No action ;
end if
if $StatusRecorded == Sending$ **then**
  Cancel the Retransmission Timer (RTX);
  Stop sending;
  Record sequence number of last sent packet;
end if
if $StatusRecorded == Waiting$ **then**
  RTOoccurred = FALSE;
  DisconnectionOccurred = TRUE;
end if

Connection()
NetworkStatus = Connected;
if $StatusRecorded == idle$ **then**
  Normal TCP behaviour;
end if
if $StatusRecorded == sending$ **then**
  Resume sending from last sent packet;
  Set RTX timer ;
end if
if $StatusRecorded == waiting$ **then**

  if $RTOoccurred == TRUE$ **then**
    Retransmit last sent packet;
    RTOoccurred = FALSE;
  else
    No action;
  end if
end if

RTO()
if $NetworkStatus == Connected$ **then**

  if $DisconnectionOccurred == TRUE$ **then**
    DisconnectionOccurred = FALSE;
    Retransmit last sent packet;
  else
    Normal TCP behaviour;
  end if
else
  ssthresh = max (ssthresh, cwnd, receiver advertised window);
  cwnd = 1;
  RTOoccurred = TRUE;
  /* No change in RTO value */
end if

AckRx()
if (new ACK) **then**
  DisconnectionOccurred = FALSE;
end if

IV. Simulations

We implemented ATCP, 3DA [10] and Freeze TCP [4] in the network simulator **ns-2.1b8a** [13]. TCP Reno is already implemented in this ns-2 distribution. We also modified **MobileIP** [7] in ns-2 for providing mobility information to the TCP Agent. Mobility of the MH is simulated by maintaining a variable, *NetworkStatus*, in TCP Agent whose value changes from *Connected* to *Disconnected* or vice versa, as determined by a *disconnection timer handler*. The disconnection timer can be configured to alternately connect and disconnect to simulate disconnection.

The simulation scenario is shown in figure 3. An FTP application simulated a large data transfer, with packet size 1000 bytes, and throughput of TCP connections was measured. For the *disconnection duration*, values ranging from 50ms to 4s were chosen. Typically smaller values occur in wireless LANs and larger values occurs in wireless WANs. The *disconnection frequency* was chosen as 10 seconds, indicating moderate to high mobility. The *round trip time* (RTT) was chosen as 5ms for representing a wireless LAN (WLAN) and 700ms for representing a wireless WAN (WWAN). The *link capacity* was taken as 10Mbps for the 5ms RTT (WLAN), and as 100Kbps for the 700ms RTT (WWAN). The capacity of both the links, i.e., from FH to BS and from BS to MH were maintained to be equal to avoid packet losses due to buffer overflow in routers. The simulations were carried out for 100s for WLAN environment and for 1000s for WWAN environment.

A. Analysis & Comparison

Observations resulting from the simulation are as follows:

- MH to FH data transfer: The 3DA and Freeze TCP approaches do not mention any actions specific to the MH being the TCP sender, hence we compared ATCP only with TCP Reno in this case. As can be seen from figures 4, 5:
  – ATCP shows more enhancement in throughput as compared to TCP Reno, as the disconnection interval in-

ATCP: FH to MH Data Transfer

DataRcvd()

```
/* Let S be the sequence number of the cumulative ACK to be sent */
if AckPending then
    if PendingAckSeqNo < S then
        Send ACK with SeqNo S-2 ;
        LastSentAck = S-2 ;
        PendingAckSeqNo = S ;
        AckPending = TRUE ;
        Set timer to invoke AckSend() after d ms;
    else
        Send two ACKs with SeqNo S;
        Cancel timer to invoke AckSend();
        AckPending = FALSE ;
    end if
else
    if LastSentAck < S then
        Send ACK with SeqNo S-2 ;
        LastSentAck = S-2 ;
        PendingAckSeqNo = S ;
        AckPending = TRUE ;
        Set timer to invoke AckSend() after d ms;
    else
        Send ACK with sequence number S;
    end if
end if
```

AckSend()

```
if NetworkStatus == connected then
    Send ACK with SeqNo = PendingAckSeqNo;
    LastSentAck = PendingAckSeqNo;
    AckPending = FALSE;
else
    Return ;
end if
```

Disconnection()

```
NetworkStatus = disconnected ;
```

Connection()

```
NetworkStatus = connected ;
if AckPending then
    Send ACK with SeqNo = PendingAckSeqNo - 1 and Advertised Window = 0;
    Send ACK with SeqNo = PendingAckSeqNo and Advertised Window = Full;
end if
```

creases. This increase in throughput for ATCP is due to the following reasons:

- TCP Reno backs off exponentially when an RTO event occurs during the disconnection. When the MH is reconnected, TCP Reno waits for its retransmission timer (RTX) to expire. ATCP does not have this "idle period" and shows better performance. Also, as the disconnection period increases, the number of RTO events increases. This results in exponentially increasing RTX values, thereby increasing the idle period for TCP Reno before it attempts retransmission.
- At each RTO event, TCP Reno decreases the slow start threshold (*ssthresh*) by half. This is undesirable if RTO has occurred while the MH is disconnected. ATCP does not exponentially decrease *ssthresh* but sets it equal to the congestion window (*cwnd* value reached at the time of disconnection. This results in ATCP attaining full window capacity more quickly than TCP Reno.

– The enhancement in throughput for ATCP over TCP Reno increases more significantly for large RTT connections. This is because connections with large RTT have correspondingly larger values of RTX, thereby increasing the idle period for TCP Reno.

– A percentage improvement of upto 40% is observed for short RTT connections, while an improvement of upto 150% is observed for long RTT connections, with long disconnection periods.

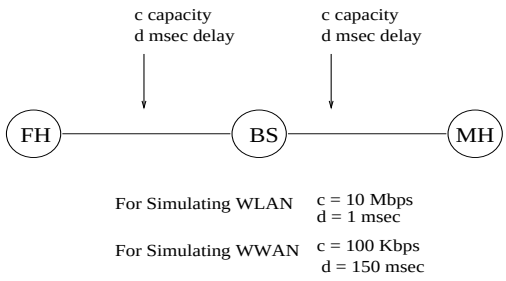

Fig. 3. Simulation Setup

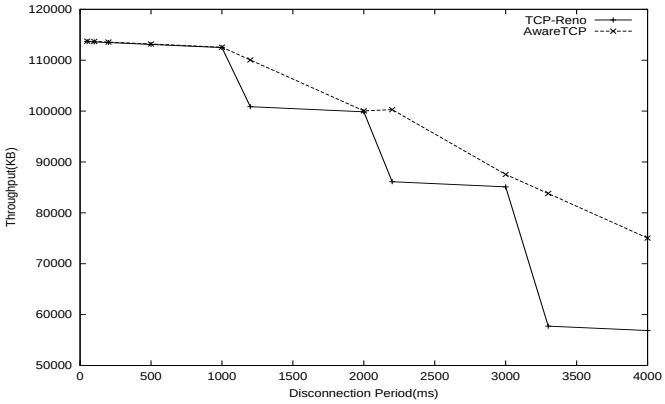

Fig. 4. MH to FH data Transfer: RTT $\approx$ 5ms

- FH to MH data transfer: Although the focus for ATCP is more on MH to FH data transfer, we also studied its performance for FH to MH data transfer. We compared ATCP with TCP Reno, 3DA and Freeze TCP. As seen from figure 6, 7:

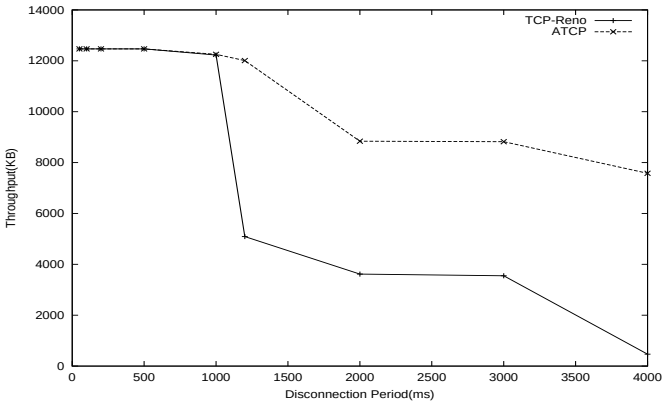

Fig. 5. MH to FH data Transfer: RTT $\approx$ 700ms

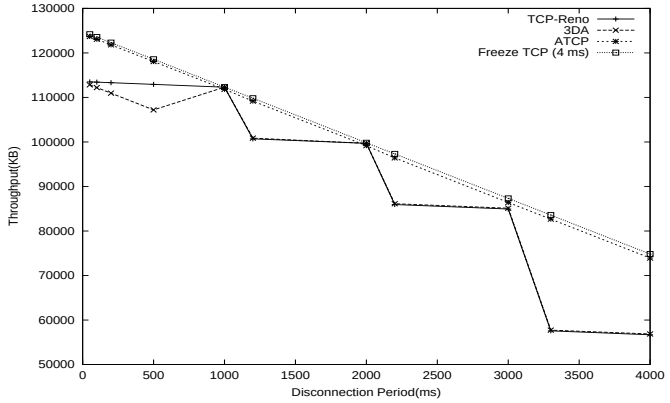

Fig. 6. FH to MH data Transfer: RTT ≈ 5ms

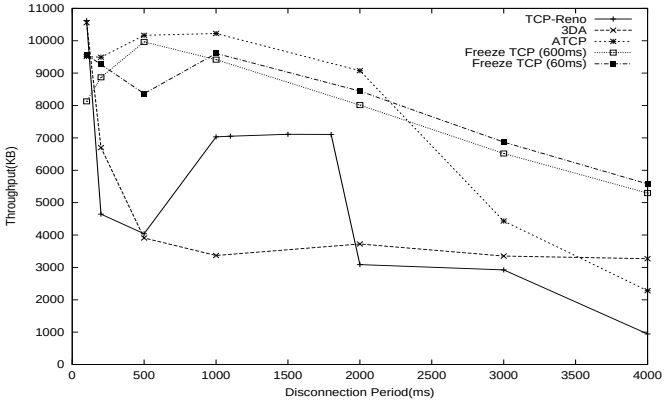

Fig. 7. FH to MH data transfer: RTT ≈ 700ms

– ATCP shows uniform improvement in throughput over TCP Reno and 3DA.
– In WLAN environments, the performance of ATCP is very close to that of Freeze TCP. ATCP and Freeze TCP showed an improvement of upto 40% over TCP Reno for long disconnection intervals. In WLAN environments, the idle period after reconnection is the primary factor in degrading the throughput rather than the reduction in the congestion window (*cwnd*). As ATCP and Freeze TCP both reduce this idle period, they both perform better than TCP Reno.
– In WWAN environment, the performance of ATCP is very close to Freeze TCP for small disconnection intervals. For disconnection intervals of upto 1000ms, both ATCP and Freeze TCP showed upto 150% improvement over TCP Reno. For longer disconnections, ATCP shows only 50% improvement whereas Freeze TCP shows 150% improvement over TCP Reno. In WWAN environments, both the idle period and the reduction in *cwnd* play significant role in degrading the throughput. ATCP is able to reduce idle period and hence performs better than TCP Reno. For small disconnection intervals where no RTO event occurs, ATCP does not change the *cwnd* value and hence achieves almost the same throughput as Freeze TCP. For long disconnection intervals, ATCP cannot prevent reduction in *cwnd* value and hence shows lesser throughput than Freeze TCP. However, Freeze TCP depends on predicting impending disconnection and its throughput was observed to be sensitive to variations in the prediction period.

V. Conclusion

We have proposed a new approach called ATCP for alleviating the degrading effect of host mobility on TCP performance. ATCP requires modification to TCP only at the MH and is optimized for data transfer from MH to FH, as well as from FH to MH. ATCP uses feedback from the network layer at the MH in terms of *disconnection* and *connection* signals, to rapidly regain the full window after the MH gets reconnected. We have compared ATCP with TCP Reno, 3DA and Freeze TCP by simulations using ns-2.

ATCP performs better than TCP Reno in both directions of data transfer. We have observed improvements of upto 40% in WLAN environments, and upto 150% in WWAN environments. The 3DA and Freeze TCP approaches do not specifically deal with MH to FH data transfer and hence ATCP is compared with them only for FH to MH data transfer. Here, ATCP performs better than TCP Reno and 3DA and its performance is generally comparable to that of Freeze TCP.

References

[1] S. Mascolo and Claudio Casetti, *TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links*, ACM SIGMOBILE 7/01 Rome Italy, ACM ISBN 1-58113-422-3/01/07, July 2001.
[2] George Xylomenos, G.C. Polyzos, Petri Mahonen and Mika Saaranen, *TCP Performance Issues over Wireless Links*, IEEE Communications Magazine, April 2001.
[3] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan, *WTCP: a reliable transport protocol for wireless wide-area networks*, Proceedings of ACM MOBICOM 99, Seattle, Washington, August 1999.
[4] T. Goff , J. Moronski, D. S. Phatak, and V. Gupta. *Freeze-TCP: A true end-to-end enhancement mechanism for mobile environments*, in INFOCOM, (Israel), 2000.
[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R.H. Katz, *A Comparison of Mechanisms for Improving TCP Performance over Wireless Links*, IEEE/ACM Transactions on Networking, December 1997.
[6] K. Brown and S. Singh *M-TCP: TCP for Mobile Cellular Networks*, ACM Computer Communications Review, vol27, no.5, 1997.
[7] C. Perkins, *RFC 2002: IP Mobility Support*, October 1996.
[8] Ajay Bakre, B.R. Badrinath *I-TCP: Indirect TCP for Mobile Hosts*, Tech Rep., Reuters university, May 1995, http://www.cs.rutgers.edu/ badri/journal/contents11.html.
[9] H. Balakrishnan, V.N.Padmanbham and R.Katz *Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks*, Wireless Networks, vol.1. no.4., Dec 1995.
[10] Ramon Caceres and Liviu Iftode, *Improving the performance of reliable transport protocol in mobile computing environments*, IEEE JSAC Special Issue on Mobile Computing Network, vol. 13, no. 5, June 1995.
[11] W. Richard Stevens *TCP/IP Illustrated, Volume 1, The Protocols*, AWL, 1994.
[12] R. Braden, *RFC 1122 Requirements for Internet Hosts - Communication Layers*, October 1989.
[13] The network simulator ns-2.1b8a/,*http://www.isi.edu/nsnam/ns/*.