Journal of Computer Science 4 (8): 680-692, 2008
ISSN 1549-3636

An Improved TCP Connection Protocol over Wireless Mobile Networks with Mobile Nodes Handoff

Maen Mahmoud Al Assaf
Department of EECS, Syracuse University, Syracuse, NY

Abstract: Problem Statement: This study proposes a TCP layer protocol (Div-TCP) that aims to give a solution for communication over wireless mobile networks that have the occurrence of handoff at anytime. It describes a topology of wireless mobile network and the handoff process as an ordinary process in that type of networks which may cause segments losses. **Approach:** Wireless mobile networks have many weaknesses related to bit error, network congestion and weak signals that cause segments losses as well as handoff process. For this reason, wireless mobile network TCP cannot distinguish between losses caused by these weaknesses or by the handoff process. So in handoff case, segments losses will trigger congestion control algorithms that reduce the TCP connection's throughput performance. **Results:** However, in addition to the previous efforts that provide different algorithms and protocols which aim to come up with different solutions and enhancements for the TCP connection in wireless mobile networks when handoff occurs, I am providing in this study Div-TCP protocol that runs the ordinary TCP algorithms with adding more operations that are concerned to handoff. These operations aim to maintain a suitable RTO at each sender to avoid triggering congestions algorithms and also to maintain the sender's throughput by establishing a TCP connection prior to the handoff occurrence. **Conclusions/Recommendations:** The protocol's discussion shows how end-to-end TCP connection became more efficient since the new Div-TCP connection after handoff will have good and already built TCP connection and the RTO is too much high at all nodes so there is no fear of a that TCP considers congestion.

Key word: TCP connection, mobile node, home agent, foreign agent, correspondent host, mobile IP, handoff, round trip time, retransmission time out, congestion control, exponential back off

INTRODUCTION

In wireless 802.11 networking, communications take different shapes that connect end to end users using at least one non-wired part of the network. Actually, there are a lot of topologies. Mobile and cellular networks are considered wireless networks that have a particular system of components and protocols. Actually, they are important at this period of time because of their important applications.

Mobile Networks may take different topologies and structures[1]. Wireless 802.11 networks with different topologies suffer from weaknesses in the level of physical and data link layer of communication such as low signal strength, other resources transmission interference and signal fading problem that causes high bit error rate. These problems contribute in disturbing the upper layers protocols especially TCP transport layer protocol.

Mobile networks can support connection to the internet as well as the telephony communication. In this study, I can consider[1] wireless networks that run infrastructure mode which is based on a Base Station (BS) that connects and routes packets from and to Mobile Nodes (MN). But for more accuracy, I will consider the mobile network (Fig. 1) that supports mobility of the MN so it can freely move among different areas of coverage. In order to support mobility, mobile network should be able to route connection from and to MN as it moves freely between different areas of coverage. So, the mobile network here has Home Network (HN) that is controlled by Home Agent (HA). Each HN has one or many MN attached to it (Subscribers) and HA controls their information, registrations and routing.

When a MN moves to another area of coverage, it leaves the HN and register with a Foreign Network FN that is controlled by Foreign Agent FA that acts as HA but in the foreign network. The sender host whether is in HN or in FN is called correspondent host CH. Every network agent is connected to the router that connects the entire network with outside. Actually, when CH establishes a TCP connection with the MN, it first establishes a TCP connection with the Network Agent

HA of that of the particular MN's Home Network (HA), then the HA establishes another TCP connection with the MN on behalf of the CH. So, the established connection between CH and MN must go through the HA which MN is currently attached with.

When the mobile node MN changes place far away, it will no longer be able to reach its Home agent or in other words it will become out the coverage area.

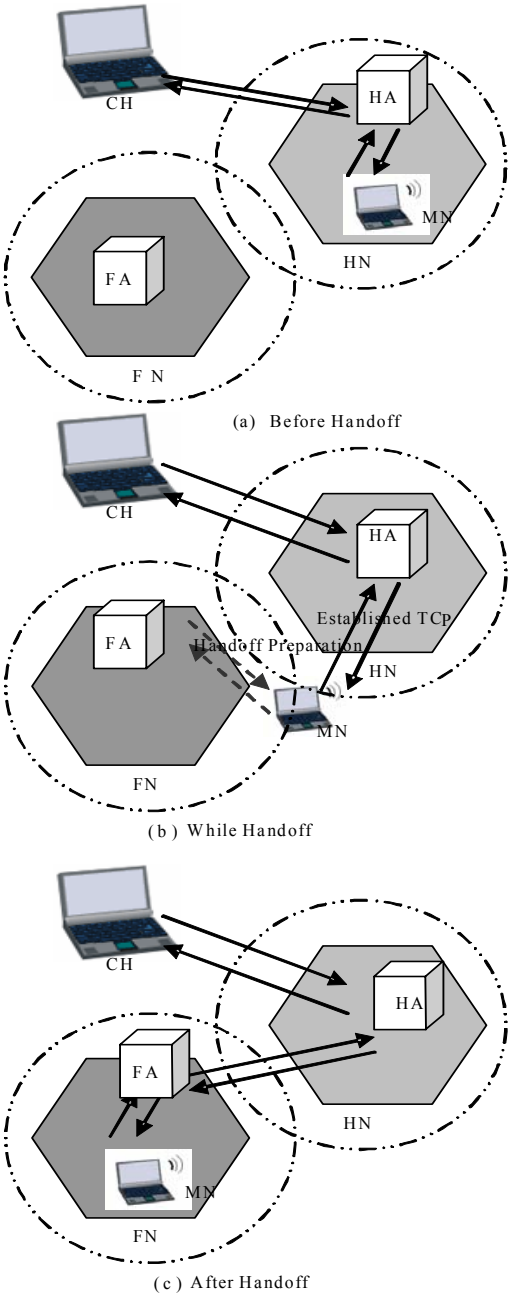

Fig. 1: Wireless mobile network topology in its different cases

Then, the HN will no longer be able to handle the connection from and to the mobile node so there should be a transfer of the connection to the next nearest foreign network FN and registration of MN with the FA in a process called Handoff. However, since the MN is related to its home network and considered as a subscriber, the connection between the CH and the MN always go through the home network agent HA even if a handoff occurs. So after handoff, the connection will be from CH to HA and then to the next foreign network's agent FA and finally to the MN. For this reason, handoff process sure disturbs network layer routing and transport layer connection as well as in many cases it may disconnect the connection especially if there is no proper protocol that handles it.

As its ability to support internet connection, mobile networks are worthy to research in its session and TCP connection because of its mentioned weaknesses especially when handoff occurs that is the scope of this study.

MATERIALS AND METHODS

Wireless connection handoff: In Mobile networking[1] Handoff occurs when MN changes its location and become under another base station or foreign network coverage during the established connection. Actually, Handoff process involves many steps including MN registration with the foreign network agent and different coordination. These processes sure take time to switch to the new network which may include disconnection at TCP transport layer. Actually, Handoff process duration is not specific. Since mobile networks can handle internet TCP/IP connection[4] TCP connection is known of its reliability connection oriented and when it is established, a stream of bytes are sent and received. So, data isn't sent separately exactly as when you make a phone call. For this reason, handoff is considerable issue.

Current work: There are many protocols, papers and projects that aim to solve transport layer and TCP connection over wireless mobile networks when handoff occurs. Kurose and Ross[1] Mobility in wireless mobile networks and the previously mentioned weaknesses in addition to handoff process should not have a remarkable impact on the higher layers protocols if the lower layers protocols heal the connection properly, so[2-3] Mobile-IP is used. But the TCP transport layer protocol is more affected than the application layer protocol because of segment loss either from congestion or by handoff and bit error. So, when a TCP connection is established between the

sender and the receiver and the sender noticed that some segments were not acknowledged by the receiver before the end of Retransmission Time Out (RTO) timer, sender's TCP will not distinguish either the segments loss is due to network congestion, due to handoff, or due to bit error and wireless weakness. Anyway, the sender's TCP will retransmit the lost segments and will apply the congestion algorithm that decreases the congestion window and throughput.

Actually, [2] Mobile-IP is considered a protocol that solves network layer problems of wireless mobile network connection. It is designed to handle mobile routing especially when handoff occurs and to re-route packets using the[1-3] care-of-address concept. So, MN should maintain its IP address granted by its HN. Also, it coordinates the MN registration process with the foreign agent when handoff takes place. In[3], Mobile-IP aims to help mobile wireless link by minimizing the number and the size of the administrative messages sent over the link. So, Mobile-IP is our layer 3 routing protocol.

In[2], TCP connection over wireless mobile networks is healed by different operational parameters like increasing window and buffer size, using large MTU and the most important is time-stamp which obtains an estimated RTT with each ACK received in order to tolerate delay without any timeout.

It is important that TCP connection over wireless mobile networks do not reach retransmitting time out RTO in order not to apply congestion control algorithms. In[5] the author used similar network topology to that I am using in this study. It agrees that long handoff and its attached process like registrations takes a period of time that causes a lot of unacknowledged TCP segments that lead TCP sender to trigger congestion control algorithms which cause exponential back off where the transmission window size (throughput) got multiplicative decrease where it returns to slow start. This will lead to a long delay before retransmitting the data packet and also will trigger congestion algorithm that decreases the TCP window size and throughput to go from the beginning slow start. It takes more time to reach the same throughput as before handoff. So, the possible solution is to modify the Mobile-IP and to allow data packets to be stored in the Foreign Agent (FA). So, when handoff occurs, the FA will forward the stored packets to the new FA. In addition, this solution is reasonable because the acknowledged packets will be removed from the old FA buffer so it will not be overloaded and also the two FAs sure are close to each other so there is no chance for disconnection while transmission. In addition, this method will help if there is no retransmission occurs. TCP retransmission time out RTO depends on RTT which depends on the link strength and the distance between the sender and the receiver. The author tested the negative effect of handoff on TCP over wireless mobile networks and compared the results with the same test using wireless and fixed network.

In[6] the authors provided a mobility management solution at transport layer that minimizes the TCP packet loss when handoff occurs based on Split Mode Approach and ensures end-to-end paradigm of TCP connection. They compared their results with TCP Reno Model. They proposed a secured authentication and micro-mobility model that use LDAP server which runs over TCP/IP and contains user profile database that makes user authentication in cooperation with RADIUS server. However, the two TCP connections are established, one between the CH and the LDAP server and the other between LDAP server and the MN. Actually, LDAP server does not send ACK to the CH unless it receives an ACK from the MN for the correspondent segments. LDAP stores data arriving from CH in SEG_TCP field as well as user profile. So, when handoff occurs, LDAP server forwards all previously buffered segments in SEG_TCP to the MN and continuously deletes the acknowledged segments. Actually, the RTT is increased and the RTO is based on the total RTT time of both TCP connections.

There are some protocols are based on Split Mode Approach to save reliability of TCP connection over wireless mobile network. Bakbe and Badbinath[7] I-TCP is proposed to handle problems related to mobility and reliability of TCP connection in mobile networks especially when handoff occurs. By this approach, the connection between MN and the base station or the network agent is wireless link but the connection between the CH or any other node with the base station or the network agent is fixed so we have two TCP connections with some improvements in the wireless I-TCP connection with the MN. Actually, the TCP fixed connection is considered to be the regular TCP. So, when MN want to connect with the CH, the base station or the network agent will establish the TCP with the CH on behalf of MN and also establish a I-TCP with the MN. When a handoff occurs, the base station or network agent will handle the connection and heal any packet loss.

Due to the unawareness of TCP connection about the network condition and the packet loses and distinguish packet loss from congestion loss and random loss. That will decrease the TCP performance when it is run over wireless networks. In[8], the authors

proposed a new TCP congestion control mechanism by router-assisted approach over wireless ad-hoc networks which is mainly based on the information feed backed from routers where the TCP sender is able to control and adjust its sending speed dynamically in order to avoid losses and congestion algorithms problems. Actually, this sure can be useful in different wireless networks including wireless mobile networks.

RESULTS

solution (Protocol) for improving TCP connection in wireless mobile networks when handoff occurs (Division TCP (Div-TCP)): In order to reach an improvement for TCP connection over wireless mobile networks, the protocol algorithm should consider the previously mentioned wireless mobile network weaknesses and its inability to distinguish between the delay that is caused by handoff, by congestion, or by bit error.

Our solution has two goals that collaborate together to cover and reduce handoff effects. Firstly, it aims to handle handoff process by avoiding the sender to reach RTO without receiving acknowledgments of the sent packets in order to avoid going back to slow start by triggering congestion algorithms so we can maintain TCP connection throughput. This is important to be maintained and its have special importance that arises after handoff takes place where the end- to-end nodes become more far to each other.

Secondly, the solution algorithm aims to establish a TCP connection between HA and the FA of the next network when the handoff process seems to happen (a little bit before handoff) in order to reach and maintain a high throughput in between. This will cover the handoff problems and affects positively upon the overall throughput of the end-to-end TCP connection.

In our protocol, we will use the previously mentioned wireless mobile network topology. We mentioned that the connection between the CH and the MN goes through the HA and if any handoff of MN happens to the next FA their will be a connection established between HA and FA. However, we call the end-to-end TCP connection that is established between the CH and the MN (Div-TCP) which is our proposed TCP solution protocol. The established Div-TCP as shown in Fig. 1 is a combination of one or more established TCP protocols between different nodes of the end-to-end TCP connection. For example, in Fig. 1a the end-to-end Div-TCP consists of the TCP connection between the CH and the HA and also the TCP connection between the HA and the MN. But in Fig. 1c which represents the MN after its handoff to the next FA, the end-to-end Div-TCP consists of the TCP connection between the CH and the HA in addition to the TCP connection between the HA and the FA and finally the TCP connection between the FA and the MN. So in both cases, the Div-TCP starts at CH and ends at MN so its parts are apply[6,7] Split Mode Approach but the whole Div-TCP is considered end-to-end TCP protocol.

It is important for each node involved in a Div-TCP connection to calculate the RTT between itself and the receiving end of the Div-TCP connection. In addition, the CH and the MN should calculate the RTT between each other along the whole path since they represent the both ends of the Div-TCP connection. Actually[4] an RTT time is the time needed for a segment to reach the destination and for its correspondent acknowledgment to be received by the sender. So, RTT is calculated at the sender side. For example, in the previous Fig. 1c after handoff occurs assuming that the MN is the receiving end, CH should calculate the RTT with the MN and also HA should calculate the RTT of its connection with the MN and also the next FA should calculate the RTT with the MN.

In order to measure the RTT between two nodes, we will use in our protocol[4,9] Time Stamp Option which is a 10-byte option in the TCP segment. Actually, the Time Stamp here is the system clock value at the segment's sending time. Initially, when a sender node establishes a TCP connection with receiver one, it sends a SYN segments that holds a Time Stamp of its system clock in the segment's options. Then, the receiver node replies with a SYN+ACK segments that holds the sender segment's Time Stamp in its options. The same thing happens while the regular TCP data transfer, each segment sent by the sender includes a Time Stamp in its options. So, when the receiver sends an acknowledgment or an accumulative acknowledgment for many received segments, it will contain the sender segment's Time Stamp in Time Stamp Echo Reply field. In other words, the receiver keeps the sender segment's Time Stamp in the acknowledgment so the sender will not be confused of not knowing for which segment an acknowledgment has been received. Then, the sender will easily calculate the RTT with the receiver after receiving the acknowledgment by subtracting the value of Time Stamp Echo Reply field from the time shown by the clock to find RTT. In[10], it considers that Time Stamp is one of the TCP extensions for high performance and should use the mnemonic RTTM (Round Trip Time

Measurement) when it is used for that to distinguish it from other Time Stamp option uses.

RTT = TSSC – TSER

Where:

RTT = Round Trip Time between sender and receiver
TSSC = The current Time Stamp of Sender clock
TSER = Time Stamp Echo Reply

In addition, the RTT is modified and recalculated continuously every acknowledgment is received. Actually, as[4] there is no need that the sender and the receiver clocks to be synchronized because the RTT calculation is based on the sender clock. In addition, the sender should not store the time the segment left because it is carried by the segment itself. Actually, we can notice that the Div-TCP between CH and the MN as they form the both ends of the TCP connection should be calculated using timestamp and should have the longest RTT.

As mentioned above, in Div-TCP, the TCP connection between the both ends CH and MN goes into multiple established TCP connections among different nodes in between. So, when the CH sends a segment to the MN, it goes through the whole path and every intermediate node stores it in its TCP buffer and forwards it through the TCP connection to the next node until it reaches the MN. Then, the MN acknowledges the received segment(s) so the acknowledgment goes back to the CH in the opposite direction. So, every intermediate node TCP waits until it receives the acknowledgment of the correspondent segment from the MN or the next nodes to clear its buffer and then forwards the acknowledgment back to the previous node and so on until it reaches the CH. For example, in Fig. 1c that shows the connection after handoff. When the CH sends a segment to the MN, it goes to the established TCP connection with the HA, then thorough the one between HA and FA then finally through the one between the FA and the MN. After that, the MN acknowledges the received segment by sending an acknowledgment back to the CH. The acknowledgment first goes to the FA then to the HA to reach finally the CH so every node clears the acknowledged segments from its buffer. For this reason, each node should consider the RTT with the receiving side of the Div-TCP connection. In addition, we can notice that the RTT increases at nodes as we move toward the sending side and decreases at nodes as we move toward the receiving side of the Div-TCP connection.

Initially in our proposed protocol, when the sender (CH) establishes a TCP connection with the receiver (MN)[6,7] Split Mode Approach is applied by establishing a TCP connection with the Home Network Agent (HA) of the MN, then HA establishes another TCP connection with the MN involved in the connection on behalf of the CH. The HA controls the both TCP connection and the end-to-end Div-TCP connection goes normally.

While Div-TCP connection is established, MN by nature moves continuously and changes place. So, the home network signal may become weak and the MN may become out of the network coverage so the MN need to handoff to another foreign network and their will be a lot of packets losses then the TCP problems start at this moment.

In order to handle these problems of our wireless mobile network, our proposed Div-TCP protocol aims to achieve the previously mentioned two goals by applying two processes before and within handoff process that we will call them Increase RTO (IncRTO) that is specialized to attain the first goal and sender TCP Buffer Division (BufDiv) that is specialized to attain the second goal. Both of them at the end collaborate together to cover and reduce handoff effects. Actually, Div-TCP applies the ordinary TCP algorithm but it adds also those two operations to cover handoff effects on the end-to-end TCP connection. It sure requires certain modifications in the TCP algorithm of those nodes that are involved in the wireless mobile network in order to understand the Div-TCP protocol which includes those two additional operations.

The first goal aims to protect the sender from reaching RTO without receiving acknowledgment. Actually[4,11] RTO is a period of time that is set at the sender side so the sender assume that the segment is lost if this period expired without receiving an acknowledgment. In this case, TCP retransmits the lost segment and apply congestion algorithms which return the window size to the beginning slow start. This will decline the throughput and sure this is something not desired. However, to reach our goal we need to increase the RTO at nodes as we move toward the sending side (IncRTO). This goal should be maintained before and after the handoff. As mentioned, this will have more significance after the handoff where the end-to-end nodes become more far so RTO that waits for acknowledgment should be more. Actually, RTO calculation is based on RTT between the two ends. It show that RTO increases as RTT increase. So in Fig. 1a

and c assuming that CH is the sending side, its RTO will be more than the other nodes thorough the Div-TCP path until we reach the MN because the RTT of the nodes decreases as we move toward the receiving side. So in our Div-TCP protocol, the IncRTO procedure is implicitly triggered at each node and the first goal is achieved. However, there are some calculations to estimate the RTO at each node of Div-TCP connection. Actually, each node should know the RTT with the receiving side of the Div-TCP connection in order to calculate the RTO. Also, as RTO calculation is effected by the value of RTT, it is recalculated as RTT changes over the time. So, RTO of sent segments may differ.

We have encountered many algorithms to calculate RTO. Peterson and Davie[11] Jacobson and Karels proposed an algorithm as a battle against congestion that determines the suitable RTO to guarantee no unnecessary retransmission of segments and no triggering congestion control mechanisms. In addition, their algorithm can be used by any end-to-end TCP protocol. It is not easy to have a given range of possible RTTs and a variation of them over the time between the two ends of the TCP connection. For this reason, TCP uses an adaptive retransmission mechanism to calculate RTO which keeps calculating the sample RTT (SampleRTT) of each segment or accumulative segments which is the difference between the segment/first segment sending time and the acknowledgment receiving time (Time Stamp) then calculating the Estimated RTT (EstimateRTT). However, if the variation among samples is small, the EstimateRTT can be more trusted and there is no need to multiply this estimate by 2 to compute the RTO. On the other hand, if the variance among samples is high, this indicates to a low trust in EstmateRTT to reach the RTO value.

It is important to state that RTO calculation is a regular process in TCP protocol. Since our protocol is based on the regular TCP protocol, RTO is implicitly calculated at every node. The following algorithm is called Jacobson and Karels Algorithm that is used to find the RTO value based on the RTT deviation where DeviationRTT is used as the following:

$$\text{Difference} = \text{SampleRTT} - \text{EstmateRTT}$$

$$\text{EstimateRTT} = \text{EstimateRTT} + (\delta \times \text{Difference})$$

$$\text{Deviation} = \text{Deviation} + \delta\,(|\text{Difference}| - \text{Deviation})$$

Where δ is a fraction between 0 and 1.It calculates the mean RTT and the variation in that mean.

Initially, EstimateRTT = SampleRTT and Deviation = 1/2×EstimateRTT.

TCP then computes the timeout value as a function of both EstimateRTT and Deviation as following:

$$\text{Timeout} = \mu \times \text{EstiamteRTT} + \Phi \times \text{Deviation}$$

Where based on experience, μ is typically set to 1 and Φ is set to 4. Thus, when the variance is too small, timeout is close to EstmateRTT; a large variance causes the deviation term to dominate the calculation.

The following Algorithm is the original one for calculating the RTO from RTT:

$$\text{EstimatedRTT} = \alpha \times \text{EstimatedRTT} + (1-\alpha) \times \text{SampleRTT}$$

$$\text{Time Out} = 2 \times \text{EstimatedRTT}$$

Where:

EstimatedRTT = Estimated Round Trip Time
SampleRTT = Sample Round Trip Time
Time Out = Retransmission Time Out (RTO)
α = Smoothing Factor between 0.8 and 0.9

Since RTO calculation is a regular process in TCP protocol where our protocol is based on that, we are not going in this paper to apply those algorithms mentioned above as we will have RTO calculated by TCP. So RTO calculation is out of this paper scope. However, the purpose of illustrating the idea of those algorithms is to show the relation between RTO and RTT when calculating RTO in TCP and how we have RTO is increased with RTT which is an important application in our proposed protocol.

The second goal aims to establish a TCP connection between CH and the FA of the next network through the HA a little bit before handoff in order to reach and maintain a high throughput which means going through slow start and become[4] increasing exponentially. In order to attain the second goal, (BufDiv) process of the Div-TCP is triggered a little bit before handoff. In this process[12], Network Agents either HA or FA periodically keep advertising ICMP messages to let any MN knows about the particular Network Agent. This is considered a regular process of Mobile-IP in wireless mobile networks. When the MN finds a new FA that has a stronger signal, it expects to handoff to save its internet accessibility.

When MN finds a new FA, it reaches to the expecting to handoff point (ExptHandoff) and BufDiv process starts here. So, ExptHandoff is the first phase of BufDiv process of Div-TCP protocol. Actually, network performance can be measured by RTT and

throughput which study as indicators of the connectivity strength between MN and HA or FA. However in this protocol, I will consider here the RTT performance measurement to measure the performance of the connection and its strength because it is affected by distance and the quality of the link between of both the sender and the receiver. So during ExptHandoff period which started after the MN encountered a FA, the MN measures the RTT between itself and the current HA and also between itself and the next FA. So, when the RTT of MN with the HA increases, it shows that the gap is increasing and there should be an absolute handoff. Actually in this protocol[13], since there is no established TCP connection up to this moment before handoff is completed between the MN and the FA, RTT is measured by sending an ICMP timestamp message from MN to FA and receiving the ICMP timestamp echo reply message that enables the MN to calculate the RTT measurement. But, in case a TCP connection is established like between MN and the HA, RTT is measured continuously by the timestamp option that is mentioned previously. Actually, in this period of time where the MN is still expecting to handoff to the FA, MN keeps sending an ICMP message to the FA as soon as it receives the previous ICMP message echo reply in order to keep calculating the RTT. So, in ExptHandoff period of time, the MN keeps monitoring and recording the total measurements of RTT (TRTT) between itself and both HA and FA.

TRTT = RTTHA+RTTFA

Where:

TRTT = MN Total RTT
RTTHA = MN's RTT with HA
RTTFA = MN's RTT with FA

Actually, RTTHA, RTTFA and TRTT are continuously recalculated during ExptHandoff period every time an acknowledgment/accumulative acknowledgment is received in case with HA and every time ICMP echo reply is received in case with FA and the new values overwrite the previous ones. However, after calculating TRTT each time, once the RTTHA in respect to TRTT becomes 38% or more and RTTFA becomes 62% or less in respect to TRTT, we call this point the end of ExptHandoff period/phase and the start of time Prepare to Handoff period/phase (PrpHandoff) which are phases of BufDiv which is an operation in our proposed protocol (Div-TCP). Actually, RTTHA percentage in respect to the TRTT increases as the MN goes far away from the HA and the RTTFA and its percentage in respect to the TRTT decreases as the MN goes toward the FA. Once MN reaches PrpHandoff, it immediately estimates a certain period of time that we will call it (EstHandoff) that when it expires, the MN will strictly disconnect with the current HA and handoff to the next FA (PrpHandoff finishes and handoff takes is done). So, EstHandoff is a value of a period of time that is calculated during the PrpHandoff phase of BufDiv operation. However, after EstHandoff period expiration, the RTTHA in respect to TRTT should be approximately about 50% or more and RTTFA in respect to TRTT should be approximately about 50% or less but not a must. But in general, the MN connectivity at that moment with the FA will be better and the RTT between them is less. In order to calculate EstHandoff period, MN in the beginning of PrpHandoff records T1which is the time that RTTHA takes to turn from 38-39% with TRTT as well as RTTFA takes to turn from 62-61% with TRTT. Also it records T2 which is the time that RTTHA takes to turn from 39-40% with TRTT as well as RTTFA takes to turn from 61-60% with TRTT. Then the average to both times (AvgEstHandoff) represents an estimated average period of time in which RTT of MN increases 1% with HA and decreases 1% with FA. However, during the BufDiv Process through its different phases (ExptHandoff , PrpHandoff) , MN keeps monitoring that there is an increase of its RTT with the HA (RTTHA) and there is a decrease of the RTT with the FA (RTTFA). Once the opposite takes place, BufDiv process is canceled and the MN will stay connected to the HA through the current Div-TCP.

AvgEstHandoff = T1+T2/2

Where:
AvgEstHandoff = Estimated time for each 1%
T1 = Time of RTTHA to turns 39% and RTTFA to turn 61% / TRTT
T2 = Time of RTTHA to turns 40% and RTTFA to turn 60% / TRTT

At this point of PrpHandoff period or phase, we still have about 10% to have that RTTHA in respect to TRTT becomes about 50% and RTTFA becomes 50% in respect to TRTT (approximately not strictly), so we can estimate the EstHandoff as the following:

EstHandoff = AvgEstHandoff×10

Where:
EstHandoff = Period of time that MN must handoff after

AvgEstHandoff = Estimated time for each 1%

We assume that MN movements and the changes of the wireless topology go constantly. However, after EstHandoff period expires, it is supposed that RTTHA in respect to TRTT becomes about 50% and RTTFA becomes 50% in respect to TRTT but it is not a must. However, the purpose of EstHandoff period is to determine a known period of time that after it expires, MN mostly will mostly have a strong signal with the FA but still it may not be an equal percentage as well as the MN may still able to hear the HA signal.

After MN calculates the EstHandoff period, it sends its value to the CH as a message. So, when CH receives Esthandoff value, it recognizes that the MN will absolutely disconnect after this period expires and the current end-to-end Div-TCP connection will no longer be established and the MN handoff will be done. Actually, when the CH receives Esthandoff stating the time left for the MN to stay under the home Network, it should be less than the Esthandoff issued from the MN because it took time to reach the CH. So, the CH calculates the actual EstHandoff (ActEstHandoff) left for the MN before handoff by subtracting the original Esthandoff time from the time it to reach from the MN to the CH which is the half of the RTT as the following:

ActEstHnadoff = EstHandoff-(RTT/2)

Where:

ActEstHandoff = The actual remaining time of MN to handoff from CH point of view
EstHandoff = The original estimated time to handoff by the MN
RTT = Round Trip Time between CH and MN

Recall that EstHandoff is calculated by MN and ActEstHandoff is calculated by CH.

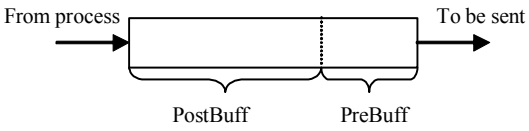

Fig. 2: CH's TCP buffer

Once the CH calculates the actual estimated time of MN's handoff during the BufDiv process (ActEstHandoff), CH immediately measures the current number of bits that are occupying its TCP buffer and also the current throughput in term of bits per second no matter how much bits of data are coming. The CH calculates the throughput of the link by sending ICMP packets over it. Susana and Rui[14] it comes as the ICMP timestamp and trace route are able to calculate the available bandwidth through the capacity and cross traffic estimation. After the CH calculates the throughput over the link, it divides its TCP buffer (Fig. 2) into two parts. I will name the first part Pre Handoff Buffer (PreBuff) and the second part Post Handoff Buffer (PostBuff). The purpose of this division is to achieve the second goal of Div-TCP which is to establish a TCP connection which is a new process from the CH to the next FA through the HA before the Handoff process takes place. This new TCP connection will be named as Interim Connection (IntCon) which as we said in the Div-TCP definition, the CH in reality establish the TCP connection with the HA and the HA establish the TCP connation with the FA on behalf of the CH. The IntCon that will endure after the completion of handoff and will bind in the new end-to end Div-TCP connection between the CH and the MN and will be the longest part of it. In the same time, the current end-to-end Div-TCP connection between the CH to the MN through the HA will stay transmitting. However, the content of the PreBuff (Fig. 2) is already located first in the queue and contains the next to be sent bits so it will stay transmitting through the current end-to-end Div-TCP connection before handoff. But the PostBuff content will be immediately allocated in the new IntCon TCP buffer and will directly start to be transmitted through the newly established IntCon before handoff process completes and will stay transmitting through the new end-to-end Div-TCP connection from CH to the MN through the FA after the completion of the handoff process (Fig. 3). Also, all the new coming bits will be buffered in the new connection's buffer.

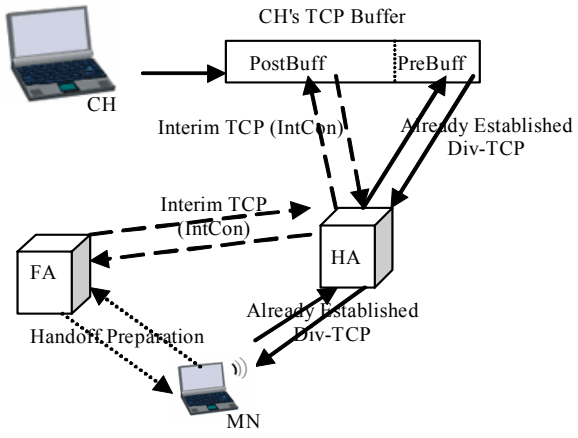

Fig. 3: Div-TCP vs. IntCon

However, in order to make this division, CH calculates the current amount of data (bits) that are allocated in its buffer and the current throughput of transmission as mentioned above. Recall that throughput is number of bits data transmission per second. Then, it finds the number of bits that should stay in the current Div-TCP connection buffer that will be transmitted in the remaining life of the current Div-TCP connection before the handoff occurs (ActEstHandoff expires). These bits will be considered as the content of the PreBuff where the rest bits will be considered as the content of the PostBuff that will be immediately transferred to the new IntCon TCP connection buffer.

PreBuffCont = ActEstHandoff×Throughput
PostBuffCont = TotalBuff-PreBuffCont

PreBuffContent = PreBuff Content in bits
ActEstHandoff = The actual remaining time of MN to handoff
Throughput = Current transmission rate in bps
PostBuffCont = PostBuff Content in bits
TotalBuff = Total buffer size in bits

As mentioned above, the purpose of establishing the interim TCP connection (IntCon) in the period of time between PrpHandoff and the completion of the handoff process is to get useful of it in order to have an existing TCP connection between the CH and the FA as a part of the near future new end-to-end Div-TCP that will have a good advertised window size and transmission rate instead of establishing it after the handoff. However, while the FA is receiving the transmitted segments from the CH through the HA over the (IntCon) in the period of time before the MN finishes the handoff process and the new end-to-end Div-TCP connection is established, it buffers and saves those segments as well as it acknowledges them for all nodes in the backward direction until the CH. In our example, FA save the received segments and send acknowledgment to the HA that will after that send acknowledgment to the CH.

After EstHandoff period ends, MN will strictly terminate the original Div-TCP connection with CH through the HA and perform[12] the handoff process and registration. So here PrpHandoff phase will finish and handoff is complete. Then, the (IntCon) will be connected with the new TCP connection that will be established between the MN and the FA after completing the handoff process (the MN becomes connected to the Foreign Network Agent (FA)) and the end to end connection (CH - HA - FA - MN) will be the new Div-TCP). Recall that all of that period between the preparation for the handoff and its completion is called PreHandoff. So now, the PreHandoff period is finished, the handoff process is complete and the MN is now attached with the FA. Here I want to summarize the BufDiv process which is an operation in our proposed protocol Div-TCP where it starts when MN encounters a FA signal so (ExptHandoff) phase of BufDiv starts and here EstHandoff period is calculated and sent to CH. Then, the second phase (PrpHandoff) will be started and here the CH divides the TCP buffer and establishes the Interim TCP connection (IntCon). Finally, (PreHandoff) is finished after EstHandoff is expired and Handoff is complete.

After the handoff is complete, the FA will forward the buffered segment that it received from the CH and will stay accepting the incoming segment traffic and acknowledging them. At that time, the MN will only acknowledge the segments that it receives to the FA not to the whole path back to the CH along the end-to-end Div-TCP. This is because the MN has just finished handoff to the FA and it is receiving the buffered segments by the FA. Those segments are already acknowledged to the CH and HA by the FA. Recall that the receiving end of Div-TCP is the one who acknowledges the received segments from the CH and this acknowledgment goes backward through the entire path and each node is acknowledged as well. But in the period of time after the handoff finishes and the buffered segments in FA are forwarded to the MN, the Div-TCP will handle a special acknowledgment policy that allows the MN to acknowledge only the FA where the FA handles the CH and the previous nodes acknowledgment. This policy will terminates only if under any circumstance the FA has forwarded the entire buffered segments to MN and it becomes able to directly forward the incoming segments without buffering them. At that time, the MN will be the one who acknowledges the received segments to the CH and all the nodes though the entire Div-TCP path.

Finally, I found that Div-TCP is a suitable name for this proposed protocol because the end-to-end Div-TCP connection between the CH and MN is divided into many TCP/Interim TCP connections between each two adjacent nodes of the connections. Also, Div-TCP protocol divides the CH TCP connection and buffer into two connections and buffers as soon as handoff is expected in BufDiv process.

DISCUSSION

Protocol Discussion: We are going to consider the topology that is illustrated in Fig. 1 to discuss our

protocol. In Fig. 1, we can see the different states of our communicating parities before, while and after handoff. At the beginning, the CH establishes an end-to-end Div-TCP connection with the MN through the HA where the MN is still in the same coverage area. The end-to-end Div-TCP is going to be the TCP connection between the CH and the HA and the other one between the HA and MN. Then, after MN handoff to the other network, the end-to-end Div-TCP is going to be through CH-HA-FA-MN.

As a regular operation in Div-TCP, RTT is continuously calculated at every node involved in the end-to-end Div-TCP connection between itself and the MN. Sure, as a particular station through out the path is close to the end MN, RTT will be less than the RTT of the next nodes. RTT is always updated at each node when it receives an acknowledgment. So let us assume at a certain period of time before handoff takes place that RTT of HA with MN is 1.5 ms, sure the RTT of CH with the MN will be more as there is more distance and assume that to be 2.5 ms. However, after handoff takes place, the values will change as MN will become more far to CH and new nodes become involves. So let us assume that at a certain period of time after handoff that RTT of FA with MN is 1.5 ms, RTT at HA to MN is 2.5 ms, and at CH to MN is 3.5. So, we relied on the fact that RTT increase with distance. But you may say that RTT not only depends on the distance but also depends on the quality of the line. But since part of the Div-TCP connection between a particular node and the MN relies on the Div-TCP connection between the next node toward the MN and the MN itself (for example Div-TCP connection between HA and MN relies on that one between FA and MN), so distance will be the main reason that controls the RTT. However as mentioned, RTTs are always recalculated and changed.

	FA	HA	CH
RTT (ms)	1.5	2.5	3.5
RTO (ms)	4.5	7.5	10.5

Fig. 4: RTO vs. RTT after handoff

When Div-TCP is established either before or after handoff, it keeps running the IncRTO operation at each node and checking its correctness. This operation leads to make RTO high enough in respect to the RTT between a particular node involved in the Div-TCP and the MN which is the end part of the connection. It is important to increase the RTO as we go far from MN toward CH to avoid congestion algorithms to be applied because the delay of acknowledgments is going to be more as the RTT increases. Actually, increasing RTO is very important especially after handoff so we need to increase the RTO of the end-to-end Div-TCP connection between CH and MN as the MN now become more far than before.

As mentioned before, the RTO calculation algorithms show that whenever the RTT is increased, RTO is also increased too. So the relationship between RTT and RTO is considered positive. In addition, calculating RTO is considered a part of the regular TCP protocol in which our protocol is based on. So the process of RTO calculation is out of this paper scope but we will give an example. The example that we will give is in [4], the mentioned RTO calculation algorithms are applied and summarized in Fig. 4. So, this clearly shows the relation between RTT and RTO as a positive relation. Actually, the given examples here are for the sake of assumption. Consider that a system calculated different values of RTO for the given RTT in Fig. 4, they should not have much variance or gap results. However, all of the mentioned in the sum shows that IncRTO operation as part of our proposed Div-TCP protocol which aims to maintain higher RTO as RTT increases is implicitly achieved.

Now let us assume that the MN started to move far away the HA and it encountered a signal from a FA, sure the Div-TCP connectivity of MN with HA will be decreasing and the Div-TCP connectivity of MN with FA will be increasing. As mentioned, RTT will be the connectivity measurement where more connectivity will have less RTT and the opposite is correct. As MN is moving away from HA to FA, a handoff will take place at the end. So, at the point where FA is encountered, BufDiv operation of Div-TCP will be triggered at MN and latter at CH and the first phase ExptHandoff will start at MN. Here, MN will keep calculating the RTTHA and RTTFA. Let us assume that at the beginning of ExptHandoff RTTHA = 35 ms and RTTFA = 65 ms. So, TRTT = 35 + 65 =100 ms. At this moment, the condition of going to PrpHandoff is not matched yet. After short period of time, the new calculation has measured that RTTHA = 38 ms and RTTFA = 62 ms, PrpHandoff now goes since the condition that RTTHA in respect of TRTT = 38% or more and the RTTFA in respect of TRTT = 62% or less. Here, the protocol will calculate (EstHandoff) period of time which a duration that when it expires, the MN will strictly handoff. To perform that, the protocol will measure an approximate time where there is a 1% increase in RTTHA in respect to TRTT and a 1% decrease in RTTFA in respect to TRTT as MN is moving toward FA and is becoming far away from HA. As mentioned previously, this involves taking the

average of two 1% increases and decreases in RTTHA and RTTFA respectively. Actually, this depends on many factors like the conditions of the cline and the way that MN moves. Let us assume that 70 ms was the period of time (T1) that RTTHA becomes = 39 ms and RTTFA = 61 ms and 75 ms was the period of time (T2) that RTTHA becomes = 40 ms and RTTFA = 60 ms as MN is moving toward FA. This will give us the value of AvgEstHandoff = (70 + 75) / 2 = 72.5 ms. So, this will lead us to EstHandoff = 72.5 × 10 = 725 ms where it is expected that the connectivity of MN with HA and FA is approximately to be equal with more transition toward FA. So after this period expires, PrpHandoff will be finished and MN will disconnect with HA and handoff to FA. Note that it is supposed that T1 and T2 to be higher than RTTHA and RTTFA since the calculation of their updated values depend on receiving acknowledgments. Fig. 5 summarizes the PrpHandoff results at MN.

EstHandoff will be sent immediately from MN to CH as soon as it is calculated by MN during PrpHandoff. CH will start now its BufDiv operation and calculates ActEstHandoff at itself side. This involves its calculation of the current RTT between itself and MN before handoff at that instance of time. This value sure will be more than RTTHA that is calculated above as the distance is more. So, assuming that RTT between CH and MN = 56 ms, ActEstHandoff = 725 – (56/2) = 697 ms which is the strictly remaining time of PrpHandoff phase and MN completely handoff from the CH's point of view.

From now until the end of PrpHandoff phase, MN has nothing to do but to wait the end of this period to perform successful handoff. But the CH now will establish the IntCon with the FA through the HA after dividing the transmitting Div-TCP buffer which will be high throughput part of the near future end-to-end Div-TCP after handoff complete to cover handoff problems.

	RTTHA(ms)	RTTFA(ms)	TRTT(ms)
	38	62	100
	39	61	100
T1(ms)	70		
	RTTHA(ms)	RTTFA(ms)	TRTT(ms)
	39	61	100
	40	60	100
T2(ms)	75		
AvgEst-Handoff(ms)	72.5		
EstHand-off(ms)	725		

Fig. 5: PrpHandoff Phase Results at MN

CH-MN RTT (ms)	56
ActEst-Handoff(ms)	697
TotalBuff(bits)	4000 bits (500 byte)
PreBuffCont(bits)	697
PostBuffCont(bits)	3303

Fig. 6: PrpHandoff Phase Results at CH

CH will immediately now calculate the total size of its TCP buffer (TotalBuff) in bits and let us assume that = 4000 bits (500 byte). Also it will calculate the current transmission throughput of the current Div-TCP established with the MN. Suppose that the current throughput = 1000 bits/sec. So, the TCP will be divided into two parts where PreBuff of size PreBuffCont = 697 / 1000 × 1000 = 697 bits and PostBuff of size PostBuffCont = 4000 – 697 = 3303 bits. So, the current Div-TCP will transmit 697 bits and it will establish IntCon connection that will transmit 3303 bits (if they are already available or they are coming from the process) to FA through HA where part/most of them will be transmitted during ActEstHandoff before the handoff completes. Fig. 6 summarizes the PrpHandoff results at CH.

Finally, the CH will now synchronize and establish the IntCon with the FA that will be an efficient part of the near future end-to-end Div-TCP and start transmitting the PosBuff content since the beginning of ActEstHandoff. TCP transmission over IntCon starts with slow start where the growth of throughput (here number of segments sent per round trip) is exponential until a threshold (sshresh) is reached where it becomes additive (increasing in 1 segment each round trip).

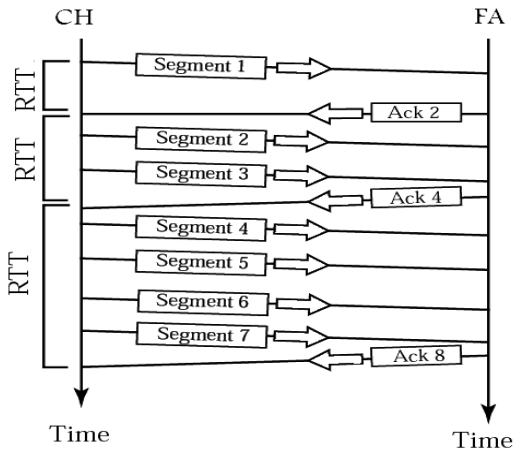

Fig. 7: TCP slow start exponential increase before threshold

Here, we are going to assume the sshresh = 16 segments. Recall that a segment is a bunch of bits. So initially, one segment will be transmitted though the IntCon in the first round trip, then 2, then 4, 8 until the threshold 16 is reached so the number of transmitted segments will be 17, 18, and so on. Each leap will be attained after one RTT between CH and FA in IntCon. Fig. 7 shows the slow start exponential increase in TCP.

Let us assume that the average of the different RTT measurements between CH and FA during ActEstHandoff is equal to 90 ms, we will have one leap every 90 ms. Let us assume that each segment holds 10 bit, so 10 – 20 – 40 – 80 – 160 – 170 and so on bits will be sent approximately after 90 ms between each one. Fig. 8[15] of slow start and congestion avoidance shows the number of segments and throughput of IntCon that will be approximately reached after the end of the ActEstHandoff where the curve is exponential until it reaches the threshold (ssthresh) so it will become additive.

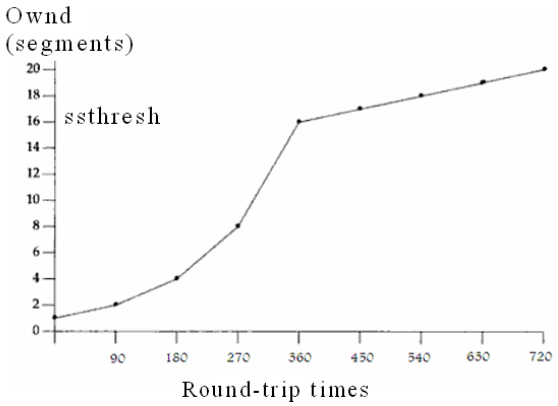

Fig. 8: Slow Start, RTT vs Transmitted Segments

We can now reach to a result from the Fig. 8 that as after ActEstHandoff which equals to 697 ms is finished, about 19 + 18 + 17 + 16 + 8 + 4 + 2 + 1 = 85 segments which equals 850 bit of the PostBuff are going to be transmitted through IntCon before the handoff completes. So, the new end-to-end Div-TCP will be efficient as it has a part with high throughput. FA will keep acknowledgment the CH until MN completes handoff, receive the buffered segments, and become able to acknowledge the CH immediately.

It is important to state that the values that were used in this discussion and were applied on the protocol algorithm may be approximate to those values when applying or simulating this protocol under a practical systems or circumstances. However, the purpose is to show that the protocol (solution) has achieved its goals. So, when this protocol is applied on practical or deployed systems, sure it is going to give at least approximates to the desired results.

CONCLUSION

TCP is a connection oriented protocol that is established between the end-to-end communicating parties for reliable communication. It handles lost packets due to the different reasons like bit error, network congestion, weak signals and handoff process. But unfortunately, it thinks that any loss is due congestion. When TCP thinks that congestion took place, it triggers congestion algorithms which return the transmission window size and throughput to the beginning slow start which is undesired event especially in case the loss is not due congestion. Handoff is a process that takes place in wireless networks that causes packet loss due to the jump of the mobile node from a network to another. This sure will make the sender TCP thinks that the loss is due congestion. For this reason, Div-TCP is a proposed solution that runs the ordinary TCP algorithm with extra two operations that are concerned to handoff process. Those operations are triggered by the Div-TCP as soon as handoff is expected to happen. IncRTO is concerned in maintaining a suitable RTO at each node involved in the end-to-end connection to avoid triggering congestions algorithms where BufDiv is concerned in maintaining the sender's throughput by establishing a TCP connection prior to the handoff occurrence.

Future work: Assume that after CH established IntCon during PrpHandoff with FA in BufDiv operation, the MN decides to return back toward the Home Network. We discussed this issue and we said that in case that MN returns toward HA, BufDiv operation will be cancelled. But after transmitting some data to the FA through the IntCon, a solution should be founded to do rollback. So the solution will be sure enabling the CH to retrieve the buffer status before establishing IntCon and to enable the FA to discard the IntCon transmission.

REFERENCES

1. Kurose, J.F. and K.W. Ross, 2004. Computer Networking: A Top-down Approach Featuring the Internet. 3rd Edn. Addison-Wesley Educational Publishers Inc., U.S.
2. Halsall, F., 2005. Computer Networking and the Internet. 5th Edn. Addison Wesley. ISBN: 0321263588.

3. Perkins, C., 2002. IP Mobility Support for IPv4", RFC-3220. http://www.ietf.org/rfc/rfc3220.txt
4. Forouzan, B.A., 2007. TCP/IP Protocol Suite International. 3rd Edn., McGraw-Hill Higher Education. New York, Printed in Singapore. ISBN: 978-007-126066
5. Fladenmuller, A. and R. De Silva, 1999. The effect of Mobile IP handoff on the performance of TCP. J Mobile Networks Appl., 4: 131-135. Doi: 10.1023/A:1019138629630
6. Rojas, O.R. and J.B. Othman, 2005. A solution to improve the TCP Performance in the presence of handoffs in Wireless IP Networks. In: Proceeding of IEEE (ICAS/ICNS 2005). pp: 13.
7. Bakbe, A. and B. Badbinath, 1995. I-TCP: Indirect TCP for Mobile hosts. In: Proceeding of 15th International Conference on Distributed Computing Systems (May 1995). pp: 136-143.
8. Yao-Nan Lien and Ho-Cheng Hsiao, 2007. A new TCP congestion control mechanism over wireless Ad Hoc networks by router assisted approach. In: Proceeding IEEE (ICDCSW'07 2007). pp: 84.
9. Clark, M.P., 2003. Data Networks, IP and the Internet: Protocols, Design and Operation. John Wiley and Sons, England, pp: 866. ISBN: 0-470-84856-1.
10. Jacobson, V. and R. Braden, 1992. D. Borman, TCP Extensions for High Performance, RFC1323. http://www.ietf.org/rfc/rfc1323.txt
11. Peterson, L.L. and B. Davie, 2003. Computer Networks-A Systems Approach. 3rd Edn., Morgan Kaufman, U.S. ISBN: 155860832X.
12. Dixit, S. and R. Prasad, 2002. Wireless IP and Building the Mobile Internet. Artech House Publishers. U.S, ISBN:158053354X.
13. Postel, J., 1981. Internet Control Message Protocol. DARPA Internet Program Protocol Specification, RFC 792. http://www.ietf.org/rfc/rfc0792.txt
14. Susana Sargento and Rui Valadas, 2006. Capacity and cross-traffic estimation of all links in a path using ICMP Timestamps. In: Proceeding IEEE (ICNICONSMCL'06), 2006. pp: 49.
15. Richard Stevens, W. and Gary R. Wright, 1994. The Protocols. Addison-Wesley, pp: 576. ISBN 0201633469, 9780201633467 http://books.google.com/books?id=-btNds68w84C