|  |
| --- |
| e-PG Pathshala logo.png  **Information Technology** |
| **Mobile Computing** |
| **Module: M-TCP** |

## **Learning Objectives**

* Limitations of I-TCP and Snooping TCP
* Introduction to M-TCP
* Architecture and TCP layer
* Functioning of M-TCP
* Advantages and limitations of M-TCP

## **Introduction**

We have seen in previous modules that that standard TCP used with fixed hosts and wired networks pose serious drop in throughput if used without modification in wireless environment. The reason is that standard TCP understands only congestion as the reason of packet loss and triggers congestion control mechanism whereas in wireless networks there can be many other reasons of packet loss like high bit error rate, frequency disconnection, improper handovers. In such a situation, applying congestion control mechanism would lead to serious drop in the throughput. Therefore there is need optimized algorithms for TCP in wireless environment. But fixed networks are already based on standard TCP and it is not possible to entirely change the entire TCP. Therefore the algorithms proposed should have the following goals.

1. It should not aim to modify the entire TCP because it is the based on which the whole of internet rely
2. The protocol should not trigger congestion control mechanism when it is not required
3. The network related problems on wireless link should be isolated from the fixed hosts

The TCP variants devised for wireless environment should adhere to these goals. In the previous module we discussed two protocols I-TCP and snooping TCP. Both are split connection protocols and divide the end-to-end TCP connection into two parts via an intermediate host. The connection between fixed host and intermediate host has standard TCP whereas optimized TCP resides on link between intermediately host and mobile host. In I-TCP the intermediately hosts act as proxy which takes packets from fixed host, gives acknowledgement and then forwards the packet to mobile host and same thing is repeated on other side. This completely loses end-to-end semantics of TCP because the packets are acknowledged even before they are delivered. The situation is worst when the mobile moves to new network. All the packets buffered for the mobile device needs to be forwarded moreover whole scheme fails when the intermediator crashes. As a revision to this scheme, snooping TCP was proposed in which the intermediator just caches the packets from fixed host towards the mobile host at the same time it snoops the flow of acknowledgements and packets header. In case of missing acknowledgements it forwards the cached packet to the mobile host. The scheme retains end-end semantics. Both I-TCP and Snooping TCP handles packet losses due to bit error rate and do not work well when packets are lost due to frequent disconnections..

In 1997, M-TCP was proposed by Brown & Singh. The protocol deals with the problem of periodic disconnection as compared to existing sol’s of that time which deals with problem of high bit error rate. The proposed solution is to send the sender into persist mode when the mobile device goes into disconnection state. This is done by shrinking the window of the sender. In this module we will discuss the M-TCP protocol in detail and how it performs in the situation of frequent disconnections.

## **M-TCP**

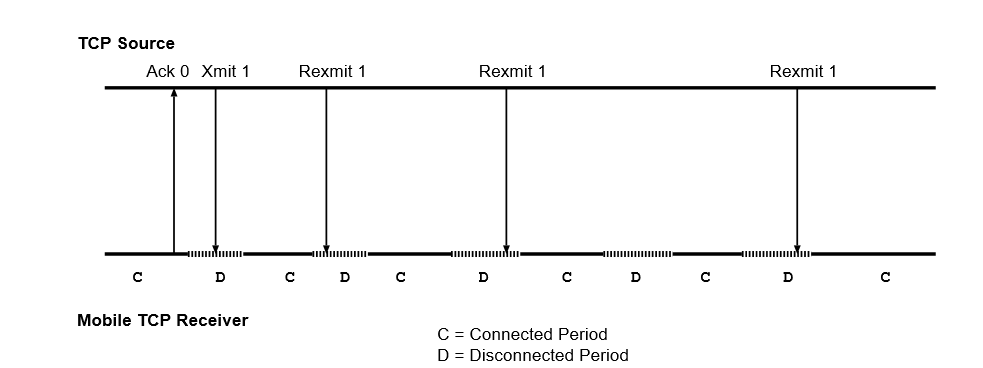
Let us see what are the reasons of frequent disconnections in mobile environment and what are the problems associated with it

## **Reasons of frequent disconnections**

* Brief blackout period (or disconnection) during handovers
* Disconnections may also be caused by physical obstacles in the environment that block radio signals, such as buildings.
* Due to load on cells, the mobile device may not receive any bandwidth for large time periods (call blocking)

## **Problems with existing solutions during disconnections**

* If the mobile is disconnected for a long period of time, the sender will invoke congestion control.
* For Snooping TCP, in the case of handoff’s the new snoop will start with empty cache and till the cache is built ,there would be no retransmission or filtering and there would be decrease in throughput. The problem of degradation would be more serious when cell sizes are small (pico or micro cell) environment
* Serial Timeout: A serial time out is a condition wherein multiple consecutive retransmissions of same segment are transmitted to mobile while it is disconnected and these retransmissions are also loss. The retransmission timer also gets doubled with every retransmission.

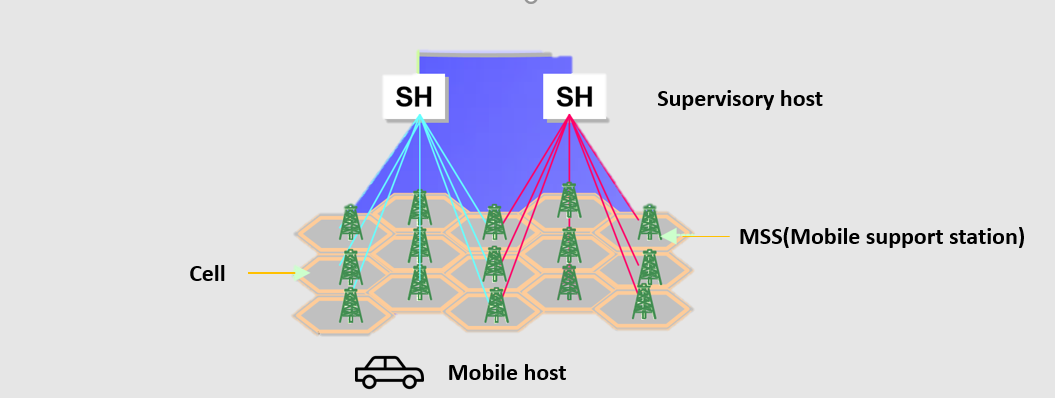


In Fig. 1 , C & D indications connection & disconnection states It can be seen that retransmission are taking place in disconnection states which again triggers doubling of timer. Hence there is no activity till 64 sec even if the device regains connection. The split connection approached using TCP on both sides of connection respond poorly to lengthy disconnections

M-TCP is aims to solve the problem of frequent disconnection, preserving end-end semantics and throughput

## **M-TCP Architecture**

* The authors of M-TCP proposed it as three-layer architecture
* Lowest level- Mobile host
* Next level- MSS (Mobile support station) is a node which controls mobile host
* Highest level- SH or a supervisory host controlling many MSS



**Advantage of having MSS**

1. MSS are cheaper devices as they serve as only connection points whereas in I-TCP & snoop protocols, sophisticated MSS are used. Hence cost of infrastructure is increases particularly in case of picocelluar networks
2. No need to transfer the states when MH roams from one cell into another controlled by same MH

## **M-TCP: Transport layer Design**

The proposed protocol divides the connection into two parts at SH supervisory host.

SH

M-TCP

TCP



SH-TCP

MH-TCP



Fixed Host

Mobile Host

* The TCP sender on fixed network uses unmodified TCP to send data to the SH while SH uses modified TCP to send data to the MH.
* The TCP client at SH receiving data from fixed host is called SH-TCP
* TCP client at SH towards MH is called M-TCP .It receives packets from SH-TCP and passes to mobile host.
* The acknowledgements are received by M-TCP from mobile host and forwarded to SH-TCP which delivers it to the fixed host

**SH-TCP client**

* SH-TCP receives a packet from fixed host
* It passes it to M-TCP client.
* It does not acknowledge maintaining end-end semantics.
* M-TCP at SH passes the packet to MH
* It receives ACK’s from Mobile host via M-TCP
* Sends the ACK’s to Fixed Host

In a situation when disconnection is there, the approach is to choke the TCP sender when mobile device is disconnected & allow the sender to transmit at full speed when the connection is regained. The mode with choked window is called persist mode. According to RFC 1112, TCP clients go to persistent mode when window size is shrunk to zero. For this purpose, a new Ack is required.SH-TCP follows a strategy to send the sender into persistent mode. It keeps one Ack from mobile host unacknowledged . The strategy is as follows:

* Suppose the window size is 100.
* 70 bytes are sent to mobile host.
* Mobile host Acks all 70 bytes
* The SH-TCP sends Ack for only 69 bytes to the sender
* When the MH goes into disconnection state after acknowledging 70 bytes, it stops sending Acks.
* When M-TCP does not receive Ack’s it assumes the mobile to be disconnected and informs SH-TCP
* It uses the last Ack for 70th byte.
* Sends it to TCP sender along with window size update
* When TCP sender receives window size update message, it goes into persistent mode
* As long as the sender is in the persistent mode, no time out occur, no exponential back off of retransmission timers occur and neither it goes into slow start

When connection is regained,

* When the MH regains the connection, it sends a greeting packet to SH.
* M-TCP receives it, passes it to SH-TCP.
* SH-TCP sends Ack to the sender and reopens it’s receive window.
* The sender leaves persist mode and begin sending data again at full speed from bytes number w+1 where w is the last acknowledged byte

The only issue with this strategy is when there only w’ bytes to be sent to MH. According to protocol, SH-TCP would send Ack’s for only W’-1. This will cause timeout at sender & trigger retransmission and after 12 retrains sender will quit. The solution to this is that when SH-TCP understands that there are no more Acks from the MH it should not allow the sender to go in persist mode. It should send the Ack for last byte.

## **Design of M-TCP**

The goal of M-TCP is to avoid serial timeouts & recover from losses due to disconnection. The modification made in this protocol is notification of wireless connectivity.

* M-TCP observes flow of Acks from MH
* If it does not receive Ack for some time, it understands mobile host is disconnected.
* M-TCP freezes all its timers. This is done to avoid congestion control mechanism at M-TCP
* It informs to SH-TCP about disconnection
* When the mobile host regains the connection, it sends a last Ack to mark the reconnection.
* M-TCP informs this to SH-TCP who opens senders transmit window
* When connection is gained, M-TCP at MH sends a specially marked Ack to M-TCP at SH, which contains Seq. no of highest byte received so far
* M-TCP unfreezes timers to resume operation

## **Advantage**

* End-End semantics is maintained because the SH does not Ack message it just forwards Ack from MH
* No buffering of packets at SH as in I-TCP so it is not necessary to forward buffers to SH
* The efficiency of the TCP connection is not degraded due to disconnections since the sender is prevented from going into exponential backoff and slow-start
* If the MH moves from the domain of one SH into the domain of another SH, the old SH does not need to forward the socket buffers (as in I-TCP).

## **Limitations**

* Errors on wireless link are also propagated to the fixed sender
* It assumes bit error rate to be very low which is not a valid assumption

## **Summary**

* M-TCP is proposed protocol which deals with packet losses due to frequent disconnections
* Idea is to choke the senders window during the period of disconnection and retransmit with full speed when connection is gained hence avoiding slow start
* The protocol divides TCP connection into two parts at supervisory host. It forwards packets but do not acknowledges them maintaining end-to-end semantics
* It has limitation of not isolating the wired and wireless connections