Architecture of TCP/IP

TCP/IP protocol consists of four layers. These layers are: Application layer, Transport layer, Internet layer, and Network access layer. The functionalities of each of these layers are discussed below:

Application Layer
Transport Layer
Internet Layer
Network Access Layer

Application Layer: The protocols at this layer are used by applications to establish communication with other applications which may possibly be running on separate hosts. Examples of application layer protocols are http,ftp, and telnet.

Transport Layer: It provides reliable end-to-end data transfer services. The term end-to-end means that the end points of a communication link are applications or processes. Therefore, sometimes protocols at this layer are also referred to as host-to-host protocols. Remember that there can be several applications or processes running on a host. Thus, to identify the end point ,it is not only the computer that needs to be identified, but also the exact process or application that would receive the message needs to be identified. This is efficiently accomplished by using the concept of port number. An application or a process specifies a port number on which it would receive a message. Once a message reaches a host, it is demultiplexed using the port number at the transport layer for delivery to the appropriate application. The transport layer provides its service by making use of the services of its lower layer protocols. This layer includes both connection oriented (TCP) and connectionless (UDP) protocols

Internet Layer: The Internet layer packs data into data packets that are technically known as IP datagrams. Each IP datagram contains source and destination address (also called IP address) --information that is used to forward the datagrams from the sender to destination through the network. The Internet layer is also responsible for routing of IP datagrams. In a nutshell, this layer manages addressing of packets and delivery of packets between networks using the IP address. The main protocols included at a Internet layer or IP (Internet Protocol), ICMP (Internet Control Message Protocol), ARP (Address Resolution Protocol), RARP (Reverse Address Resolution Protocol) and IGMP (Internet Group Management Protocol).

Network Access Layer: The functions of this protocol layer include encoding data and transmitting at the signaling determined by the physical layer. It also provides error detection and packet framing functionalities. The functionalities of this layer actually consists of the functionalities of the two lower most layers of the ISO or OSI protocol suite, namely data link and physical layers. The data link layer protocols help deliver data packets by making use of physical layer protocols. A few popular data link layer protocols are Ethernet, Token, Ring, FDDI, and X.25. Ethernet is possibly the most common data link layer protocol. The physical layer defines how data is physically sent through the network, including how bits are electrically or optically signaled by hardware devices that interface with a network medium, such as coaxial cable, optical fibre, or twisted pair of copper wires.

An Overview of the Operation of TCP

When a client-server application on hosts that are wide apart, data transmission between the client and the server may span multiple networks. These networks are called sub-networks. For data routine, the internet protocol (IP) requires that each host in the network should have a unique address. Identification of hosts is not enough for data delivery, the packets must be forwarded to the exact application (or to a process in an application) requiring the packet. Within each host, every process is identified by a port number based on which the TCP can deliver data/information to each relevant process.

Usually a message in the form of a block of data is passed to TCP by the sending application

The TCP breaks it into many small parts and attaches certain control information (called TCP header) to each small part. Each small part of the data along with the TCP is called a segment.
The Structure of a TCP segment

TCP Header	Portion of Data

Control Information

The TCP header includes several items of information including the following:
i. Destination Port
ii. Checksum
iii. Sequence number

IP Datagram

AN IP packet is also called a datagram. A datagram is of variable length whose size can be up to 65536 bytes. It has two fields, namely header and data. The structure of an IP datagram is

Version	HLen	Service	Total Length
Identification		Flags	Fragment Offset
Time to Live	Protocol	Header Checksum	
Source Address			
Destination Address			

IP Datagram Structure

Version (ver): The IP version number is defined in this field, e.g. IPV4 or IPV6.
Header Length (Hlen): It defines the header length as multiples of four bytes.
Service type: It has bits that define the priority of the datagram itself.
Total length: This field is allotted 16 bits to define the length of IP datagram.
Identification: It is mainly used to identify fragmentation that belongs to different networks. 16 bits are allotted for this job.
Flags: It deals with fragmentation of the data.
Fragmentation offset: It is a pointer to the offset of the data in the original datagram.
Time to live: This field is used to define the total number of hops that a datagram has to travel before discarding the operation.
Protocol: This field has 16-bits. It defines which upper layer protocol data is encapsulated at the time, say, for eg TCP or UDP or ICMP, etc,.
Header checksum: It has a 16 bit field to check the integrity of the packets.
Source address: It is a 4 byte (4 * 8 = 32) internet address to define the original source.
Destination address: it is a 4 byte (4 * 8 = 32) internet address to identify the destination of the datagram. The IP datagrams are sent to the link layer and become ready for transmission to the first sub-network in the path to its destination. The IP datagrams are also called packets.

Port address

In a client-server application, the client and the server programs are often located on different host machines. The client program usually uses a temporary port number and the server program uses a well-known (or permanent) port number. These port numbers are used for identification of the application. A few well-known ports used by some of the popular TCP/IP protocols and by user applications are given in Table.

A Few Commonly Used Well-Known Port Numbers

Protocol	**Port**
TELNET	23
SMTP	25
RPC	111
DNS	53

Data Encapsulation

When the TCP segments are handed over to Internet Protocol (IP Layer), this layer appends an IP header containing relevant information. A segment after this additional control data is added, is called an IP datagram .The network access layer also appends its own header and now the segment becomes known as a frame/packet. The packet header includes important information such as the following:

I. Facilities requests (such as priority)
II. Destination sub-network address.

Adaptation of TCP Window

The TCP primarily deploys a flow control technique to control congestion in a network. Traffic congestion occurs when the rate at which data can be injected by the network. A flow control technique helps adapt the rate of data transmission by the TCP at the sending host end. The flow control technique helps to prevent the buildup of congestion in the network and at the same time helps to prevent buffer overrun for slow receivers.

If data transmissions occur at a much faster rate than what the network infrastructure can comfortably support, then data packets get buildup at the routers. When the buffer at routers start to overflow, the packet start getting lost. Additionally, if data transmissions by a sender takes place at a faster rate than what a slower receiver can handle, then the receiver's buffer start to get flooded and hence the packet get lost. TCP handles both these causes of packet loss by reducing the rate at which data is transmitted at the sender's end. Thus, a receiver uses the flow control mechanism to restrict how fast a sender can transmit. However, to provide an acceptably fast data transmission service, once congestion disappears the transmission rate at the sender's end needs to be increased to a suitable value. Thus, we can say that the flow control technique helps TCP dynamically adjust the transmission rate at the sender's end, reducing the transmission rate as congestion starts to develop and increasing it as congestion starts to disappear. The flow control mechanical deployed by TCP (called the sliding window protocol) is primarily based on the concepts of congestion window and advertised window. We can brief describe these techniques.

When a sender starts to send data packets, the receiver indicates an advertised window (or receiver window) to the server while sending acknowledgments. The advertised window is usually set equal to the size of the receive buffer at the receiver. The sender uses the advertised window size obtained from the receiver to determine the maximum amount of data that it can transmit to the receiver without causing buffer overflow at the receiver. In other words, to prevent buffer overflow at the receiver, the data packets transmitted by a sender without having received acknowledgement for them should not exceed the size of buffer available at the receiving end.

For each segment sent, a sending host expects to receive acknowledgements. A congestion window indicates the maximum number of segments that can be outstanding without the receipt of the corresponding acknowledgment before the TCP at the sender's end pauses transmitting and waits for an acknowledgement to arrive. The TCP at the sender's end pauses if the number of segments for which the acknowledgement is outstanding becomes equal to the congestion window. A sender sets the congestion window size to 1 and keeps on increasing it until duplicate acknowledgement are received or until the number of outstanding packets becomes equal to the size of advertised window.

Upon receipt of an acknowledgement, TCP detects packet loss using Retransmission timer out (RTO) and duplicate acknowledgement. After transmitting a segment, a TCP sender sets the retransmission timer for only one packet. If an acknowledgement for the packet is not received before the timer goes off, the packet is assumed to be lost. Again the TCP sender assumes that a packet loss has occurred if it receives three duplicate acknowledgements consecutively. In TCP, when a receiver does not get a packet that it expects in a sequence but gets an out if order packet, it considers that the expected packet might have got lost and it indicates this to the sender by transmitting an acknowledgement for the last packet that was received in order. Thus, three duplicate acknowledgement are also generated if a packet is delivered at least three places beyond it's in- sequence location.

In wired networks, packet losses primarily occur on account of congestion encountered in the transmission path. However, in a wireless environment packet losses can also occur due to mobility and channel error. In wired networks, bit errors are rare. On the other hand, wireless network are vulnerable to noise. Noise can cause intermittent bit errors. Further, there can be intermittent disconnection due to fading and also due to obstructions that may be encountered by a mobile host. Further, packets may get lost during hand off. An intermittent disconnection may cause the TCP at the sender's end to timer out for an acknowledgement and cause it to retransmit. This would cause unnecessary retransmissions to occur, even though the packet may buffered at a router. Also, various additional causes of packet loss can result in a high rate of packet loss in a mobile wireless networks. These losses would be interpreted by TCP as symptoms of congestion and force it to operate in slow start. This would cause the network to operate inefficiency and result in unacceptable slow data transmission.

Adaptive transmission control mechanism

When many packets are transmitted to a single receiver and the rate with which these packets are transmitted is higher than the processing rate of the destination host or an intermediate router, the buffers of the router (or the destination as the case may be) get filled quickly. This results in dropping packets at the affected router or the destination. When a sender realizes that some packets might have been drooped based on acknowledgements not arriving before timeout or based on the receipt of three duplicate acknowledgements, possibly due to congestion, it tries to retransmit the missed packets. Though this mechanism may overcome packet losses, it does not resolve the congestion problem. The congestion problem is resolved through an adaptive transmission control mechanism called flow control.

Improvement in TCP Performance

TCP was designed for traditional wired networks. If used as it is, a few shortcomings became noticeable. TCP has been extended to work efficiently in a mobile environment.

Traditional Networks

In the wired networks, packet losses are primarily attributable to congestions that get built-up in the network. To reduce congestion, TCP invokes the congestion control mechanisms. Congestion control is primarily achieved by reducing the transmission window, which in turn results in slower data transfer. The important mechanisms used by TCP for improving (TCP-Reno model) performance are

Slow Start

The slow –start mechanism is used when a TCP session is started. Instead of starting transmission at a fixed transmission window size, the transmission is started at the lowest window size and then doubled after each successful transmission. The rate of doubling is tied to the rate at which acknowledgements are received. Thus, the doubling of window size occurs at every round trip time (RTT).RTT is the time that elapses between transmission of a segment by a sender and the reception of the corresponding acknowledgement. If congestion is detected (indicated by duplicate acknowledgements), the transmission window size is reduced to half of its current size and the congestion avoidance process starts. We can argue that the rate of doubling and reduction is nothing but a binary search technique deployed to

The slow-start process begins by the sender setting the transmission window size to one, i.e., transmitting one segment to the receiver. The sender does not transmit the next segment until it receives an acknowledgement for transmission of the preceding segment. Once the acknowledgement is received, the sender becomes sure that the congestion window (network capacity) is at least one segment. To determine the exact congestion window size, the sender doubles the transmission window size. It transmits two segments and after arrival of the two corresponding acknowledgements, it again increases the transmission window size by two and sets it equal to four, and so on. Increments that occur to the size of the congestion window are, thus, exponential. A congestion window is doubled every time the acknowledgements arrive smoothly. This exponential growth of congestion window stops at the congestion threshold.

Congestion Avoidance

The congestion avoidance algorithm starts where the slow start stops. Once the congestion window reaches the congestion threshold level, then after that if an acknowledgement is received the window size is increased linearly, i.e., the window size doubling is avoided. From this point, the TCP increases its transmission rate linearly by adding one additional packet to its window at each transmission time. If congestion is detected at any point, the TCP reduces its transmission rate to half the previous value. Thus the TCP seesaws its way to the right transmission rate. Clearly, this scheme is less aggressive than the slow-start phase (note the linear increase against the exponential growth in slow start).

Fast Retransmit/Fast Recovery

Usually, a sender initiates a timer after transmitting a packet and sets the timeout value (RTO). RTO is calculated based on RTT. The sender waits for an acknowledgement of a transmitted packet from the receiver until the timer expires. When the timer expires, it retransmits the packet. However, there exists another situation under which a sender can retransmit a packet. This mechanism is called fast retransmission. In this, the retransmission is not triggered by a timer, it is rather triggered by the receipt of three duplicate copies of an acknowledgement for a packet received from the sender. Since duplicate acknowledgements also arise when a segment is received out of order, the sender waits for three copies of acknowledgements for the same packet. This is taken by the sender as the confirmed indication of a missed packet for starting to retransmit the particular packet. When retransmission occurs, the congestion window size is reduced by half. For example, if the current congestion window size is four segments, then it is set to two segments. Once the lost segment has been retransmitted, TCP tries to maintain the current data transmission rate by not going back to slow start. This is called fast recovery. In fast recovery, the congestion window size is incremented by three since the retransmission occurred after the third duplicate acknowledgement. This is considered to be the indication that three packets would have been successfully buffered at the receiver end. Thus, in fast recovery, compensation for the segments that have already been received by the receiver is carried out. If the acknowledgements are received smoothly, it is considered to be the indication that there is no congestion.

TCP Congestion control Algorithms

A few popular TCP Congestion control algorithms are the following :

TCP Tahoe

TCP new Reno

TCP SACK

TCP Vegas

TCP Tahoe

TCP Tahoe is possibly the most widely used TCP congestion control mechanism. The main processes of this algorithm include: Slow-start, Congestion avoidance and Fast retransmit.

The Slow start and the Congestion Avoidance mechanisms target to maintain an optimal size of the congestion window. Slow start procedure is the first to be triggered each time a packet loss is detected. In the Slow start mechanism, there is an exponential increase of the congestion window size with the

objective of achieving optimum transfer rate. On the other hand, the congestion window size increases very slowly in the Congestion avoidance phase to prevent packet losses as long as possible.
A sender treats a packet to be lost, if it does not receive an acknowledgement for it before the time out period. It assumes that congestion has taken place, and starts the congestion avoidance mechanism. For congestion avoidance, Tahoe uses 'Additive Increase Multiplicative Decrease'. In this, it takes half of the current congestion window size as a threshold value. It then sets the congestion window to one and starts slow start until it reaches linearly until it encounters a packet loss.
Whenever, three duplicate acknowledgements for the packets are received the Congestion avoidance algorithm stops and the Fast retransmit algorithm is invoked. In doing this, it does not wait for the retransmission timer to expire before retransmitting the lost packet. It is obvious that for duplicate acknowledgments for the packet to occur, duplicate delivery for the packet must have taken place. In TCP, reliable delivery is implemented through the use of 32 bit sequence numbers for each packet. The value in the ACK field is the next sequence number that the receiver expects to receive from the sender. Since fast retransmit does not wait for the timeout, it speeds up transmission. It may be noted that is the both cases the threshold are used followed by the dividing congestion window size by two or minimum.
In case of data packet losses or out of order packets, TCP Tahoe activates congestion control algorithm. TCP retransmits the lost data packets and shrinks its congestion window size. The congestion losses are typically bursty in nature (many packets are lost at a time) thus decreasing the data transmitted over the connection maybe appropriate to recover from the losses.
Out of order packets could arise due to the specific load balancing mechanism deployed or the specific routing protocols that implement the multi-path routing approaches. It may be noted that even though the packets reach the destination in an out of order manner, but all the packets are received by the receiver. However an issue that arises with TCP, Tahoe is that it retransmits the out of order packets and triggers its congestion control algorithm unnecessarily. This leads to unnecessary reduction of transmission rate reduction over the transmission link. In this case, the reliability of TCP results in wastage of available bandwidth utilization and also needs more energy to increase its congestion window size.

TCP New Reno
TCP New Reno sends partial acknowledgements that indicate to the TCP sender about the sequence number of a segment that has been lost. This process helps in minimizing packet retransmissions over the connection resulting in improved bandwidth utilization.
TCP Reno retransmits all the data packets within the same segment and enter the Slow start phase that leads to unnecessary data packets retransmission. During fast recovery of TCP New Reno, partial acknowledgment received by the sender informs it about the exact packet that was lost, which it retransmits. In a single window, when multiple packet loss occurs, TCP New Reno invoked and lost packet is transmitted without waiting for the retransmission timer to expire. Also this technique is able to handle the issue of multiple packet losses within the same congestion window. TCP New Reno works in fast retransmit/fast recovery mode until retransmission of lost data packets is complete.
Multiple data packets losses is common in mobile wireless environments due to factors such as noise and poor signal strengths. Since TCP New Reno can effectively handle multiple packer losses, it is better suited for mobile wireless applications as compared to TCP Reno. Moreover in TCP New Reno, the sender retransmits only one packer per Round Trip Time for partial acknowledgment received from the receiver end.

TCP SACK
TCP Reno avoids waiting for the timeout delay to occur before retransmitting a packet. It aggressively retransmits after three duplicate ACKs are received. However, it may cause wasteful retransmissions of the packets since the packets might have already been received by the receiver by that time. This situation may occur, if some out of order packet reach slightly later than other packets.
A selective acknowledgement (SACK) facility enables the receivers to indicate the out of order (unsequenced) packets it receives. For this, the SACK choice is used within an acknowledgement. This choice prevents retransmission of entire window and only the lost packets are resent quickly. The SACK

option may not affect the existing congestion control algorithms. Also it retains the characteristics of TCP Tahoe and TCP Reno in case of out of order packets and so on. The main difference between the TCP SACK and the TCP Reno is the when multiple packets are lost from the window, TCP SACK retransmits only the lost packets. As an example, suppose a TCP SACK sender sends packets from 1000-4000 with 500 packets at a time. The SACK option of TCP will acknowledge the packets that have correctly been received and the sender will deduce the lost packets from the received SACK coming from the receiver side.
This technique does result in improved performance in sending/receiving of data packets over a network. However, it suffers from the following performance penalty. When there is a packet loss, in order to the sender to list the sequence number of packets that have been transmitted. On the sender side, the sender takes significant time to lost the sequence number of all the transmitted packets. But the calculations required for listing procedure causes performance degradation of the network.

TCP Vegas
The TCP Vegas is an important congestion control technique. TCP Vegas extends the retransmission mechanism of TCP Reno and calculates Round Trip Time (RTT) of each sent packer by using a fine grained clock. The retransmission timeout value is calculated each time there is a change of RTT. After receiving a duplicate acknowledgment, Vegas observes whether the difference between the exiting time and recorded timestamp for the first unacknowledged segment is greater than the timeout value , then it immediately resends the segment without any delay and not waiting for three duplicate acknowledgements.
The central idea of TCP Vegas is to calculate the retransmission time, and from that inferring the available bandwidth of the network. This is in contrast to TCP Reno that tries to increase the transmission until it overshoots the bandwidth. The available bandwidth and the actual bandwidth are computed as follows. First, Base RTT is set to the minimum of all measured RTT of the first transmitted segment of the transmission, before any router queue increases due to traffic. Based on this, the available bandwidth is computed. Next, TCP Vegas calculates the 'Actual' sending rate. This is done by recording the transmission time for a certain segment and recording how many bytes are transmitted between the times that the segment is sent and its acknowledgment is received. Based on the available bandwidth and the actual bandwidth, the congestion window size is determined. The congestion window size is increased, only if the available bandwidth is more than the actual bandwidth. The congestion window size is reduced , if the actual bandwidth is more than the available bandwidth.
This TCP extension results in improved throughput because the lost packets are identified quickly as compared to other TCP version sand ensures the recovery of lost packets. However, if the evaluation of the RTT values are wrong, then TCP Vegas performs poorly. For example, RTT may change when packets are forwarded along an alternate route in the network. In the case, if the measured RTT value increases, TCP would not be able to distinguish if the RTT increase is due to congestion or due to route changes. This way, TCP will decrease its data transmission rate interpreting that the increase is due to the onset of congestion. This action can lead to wastage of bandwidth, as transmission is inhibited which could otherwise have successfully taken place. Also, significant computational complexity is added for computation of RTT and timeout calculations on reception of each acknowledgment.

TCP in Mobile Networks
In Internet, TCP is the de facto standard transport protocol. It has been remarkably successful in supporting the diverse applications which drive the Internet's popularity. A few of such applications are access to information hosted across various websites, file transfer and email. The performance of TCP is considered satisfactory for wired networks, but it suffers from serious performance degradation in wireless mobile networks. Wired networks are significantly different from wireless mobile networks. The main differences are much lower bandwidth fluctuations with time and also as a mobile host moves, higher delay, and intermittent disconnections, high bit error rate, and poor link reliability. An implication of these differences is that packet losses are no longer only due to network congestion, they may well be also due to intermittent link failures, bit errors due to noise, or handoffs between two cells.Therefore, the

traditional TCP assumptions are not valid in mobile (wireless) environments. This leads to poor performance of TCP in mixed wired-wireless environments.
Several modifications at the transport layer have been proposed and studied in recent years to deal with the problems. To understand these works, we need to first consider single-hop wireless networks (such as wireless LANs).Based on this, we shall discuss multi-hop wireless networks (such as mobile ad hoc networks).A few important mechanisms used to improve TCP performance over mobile wireless networks are discussed below.

TCP in Single-hop Wireless Networks

I-TCP: Indirect TCP (I-TCP) protocol was proposed by Bakre and Badrinath.
It segments the connection between the fixed host and the mobile host into two different connections: the wired part and the wireless part .The wired connection exists between the fixed host (FH) and the base station (BS) and the wireless part connection exists between the BS and the mobile host (MH) .Thus, the base station maintains two separate TCP connections: one over the fixed network and the other over the wireless link. The wireless link usually has poor quality communication, but it gets hidden from the fixed network at the BS. When a packet is sent by FH to MH, the packet is received by BS first and then BS transmits it to the MH over the whole connection, information and responsibilities that are with the current BS are transferred to the new BS.
The advantage of the split connection approach of I-TCP is that it does not need any changes to be made to the standard TCP protocol. By partitioning errors in the wireless part would not propagate into the fixed network, thereby effectively achieving an increase in bandwidth over the fixed network .An important disadvantage of this scheme is that I-TCP does not maintain the semantics of TCP as the FH gets the acknowledgement before the packet is delivered at MH.I-TCP does not maintain the end-to-end semantics of and TCP assumes that the application layer would ensure reliability.

Fast Retransmission: This approach was suggested by Caceres et al to overcome the delay in transmissions caused to intermittent disconnections such as those that occurs when a mobile host (MH) moves to a foreign agent (FA) during a TCP communication. As already discussed, TCP transmission behaviour after a disruption depends on its duration. The extremely short disruptions (lasting for a time less than RTO) would appears as short bursts of packet losses. In this case, the TCP would retransmits those packets for which the timeout occurs and recovers them without slow-start. However for long disruptions (lasting for a time much greater than RTO), TCP resorts to slow-start. This results in inefficiency. As soon as a mobile host registers at a foreign agent, it starts sending duplicate acknowledgements. As in standard with TCP, three consecutives acknowledgements for the same TCP segment are inferred as a packet loss acknowledgements by the host-end TCP, and it is also inferred that the connection is live, thereby causing the fast retransmit behaviour of TCP.
The advantage of this scheme is that it reduces the time for the MH to get reconnected, otherwise FH would wait for RTO unnecessarily. The disadvantage of this approach is that it does not propose a general approach for TCP communication in mobile wireless networks. For example, it does not address the specific error characteristics of the wireless medium.

Snooping TCP(S-TCP): Balkrishnan et al proposed a protocol that improves TCP performance by modifying the software at the base station while preserving the end-to-end TCP semantic. The modified software at the base station is known as *snoop*. It monitors every packet that passes through the TCP connection in both directions that is from MN to FH and vice versa. It buffers the TCP segments close to the MH. When congestion is detected during sending of packets from the FH to MH in the form of a duplicate acknowledgement or timeout, it locally retransmits the packets to MH if it has buffered the packet and hides the duplicate acknowledgement.
An advantage of snooping TCP is that it maintains the TCP semantics by hiding the duplicate acknowledgements for the lost TCP segment and re-sends the packets locally. However, it also suffers from higher overheads, incurred when MH moves from its current BS to a new BS, the packet buffered at the current BS need not transferred to the new BS.

Mobile TCP (M-TCP)
The protocol was suggested by Kevin Brown et al. In mobile wireless networks, users would badly suffer from the problems of unacceptable delays in TCP communications and frequent disconnections caused by events such as signal fades, lack of bandwidth , handoff, unless these are explicitly handled by the protocol. The M-TCP protocol tries to avoid the sender window from shrinking or reverting to slow-start when bit errors sender window causes a packet loss, as in attempted in I-TCP and snooping TCP. In this protocol, as in I-TCP, the TCP connection between the fixed host and the mobile host is segmented into wired and wireless parts- the wired part connection between the fixed host (FH) and the supervisory host (SH) and the wireless part connection between the SH and the mobile host (MH). Many MHs are connected to SH through several base stations is shown. The SH supervises all the packets transmitted to MH and the acknowledgement sent by MH. It is also serves as an interface between FH to MH.
When a packet is sent to FH by MH using SH, the wired part uses the normal unmodified TCP and the wireless part uses modified version of TCP known as M-TCP to deliver data to MH. This packet is acknowledged only when the MH receives the packet. Thus, it maintains the TCP semantics, unlike the I-TCP. In case the acknowledgement is not received by FH, SH decides that MN is disconnected and sets the sender FH window size to zero. This prevents re-transmission. When SH notices that the MH is connected, it sets the full window size of the sender FH. When MH moves from its current SH region to a new SH region, a state transfer takes places, so that the new SH can maintain TCP connection between FH and MH.

Freeze-TCP: The basic idea in this scheme is to *"freeze"* the TCP senders' streams a little before a disconnection is to occur. This is done by artificially sending a "Zero window Advertisement "informing the sender that the receiver cannot receive data at the moment. When the sender resumes its connectivity, the receiver can unfreeze the sender by sending the value of its actual receive window. An interesting advantage of this proposal is the avoidance of the slow-start period upon re-establishment of connectivity. The freeze-TCP method has the advantage that it does not require the involvement of the intermediate nodes and hence it can be used if the IP payload is encrypted. Therefore, this method offers a promising approach for use in Virtual Private Networks (VPNs).

TCP in Multi-hop Wireless Networks
The TCP-F (TCP feedback) protocol has been proposed for extending TCP to multiple-hop networks. In a mobile ad hoc network, a sender MH sends a packet to destination MH through the intermediate MH, since all the nodes of networks are MH. Again the MHs are free to move arbitrarily which changes the network topology unpredictably. If the normal TCP runs over this network, there may be significant performance degradation at the transport layer. This happens because the normal TCP is unable to distinguish between the packets loss resulting from link failure and packet loss due to congestion on the network. As a result, the normal TCP invokes the congestion control mechanism, even if a packet loss occurs due to link failure. When this happens, TCP waits for a longer time to retransmit the lost packet and this slows down the rate of transmission. The TCP-F proposed by Chandran et al addresses the problem caused by link failures due to mobility by performing a freezing action and limiting the re-transmissions.

Consider that a source MH is sending packets to a destination MH. As soon as an intermediate MH detects the disruption of route due to mobility of MH along that route, it sends a route failure notification (RFN) packet to the source MH and records that event. Each intermediate MH that receives the RFN packet invalidates the particular route and prevents the incoming packets intended for the destination MH passing through that route. If the intermediate node MH knows of an alternate route to destination MH, this alternative route can now be used to support further communication and the RFN is discarded.Otherwise, the intermediate MH propagates the RFN towards the source MH. On receiving the RFN, the source MH completely stops sending further packets (new OR transmission) then it marks all the existing timers as invalid and freezes the send windows. The source MH remains in this state until it is notified of the restoration of the restoration of the route through route re-establishment notification (RRN) packets.