3. Wireless TCP

3.1 Performance Evaluation of TCP in Wireless Environment

TCP versions have undergone modifications in their congestion control algorithm assuming only the wired environment. To strengthen the reliability, TCP considers all the losses are caused by congestion in a fix wired network. The network congestion and server overload were considered as the main reason for the packet losses. To handle the congested environment, TCP reduces its transmission flow drastically. This assumption is not true for the wireless networks and in wireless channels (i.e. terrestrial link) with high BERs; as TCP packets may frequently be lost (discarded) due to corruption rather than congestion. Identical treatment to losses due to corruption can unnecessarily degrade the performance of TCP.

3.1.1 TCP Performance Issues

To recover from the loss, Baseline TCP has to wait for the Retransmission Time Out (RTO) and also cwnd is reset to one. Tahoe TCP uses fast retransmission to retransmit lost packet but it also sets cwnd to 1. Reno TCP enters fast recovery after getting three duplicate acknowledgements but still unable to deal with multiple packet losses in a single window of data. New Reno deals with multiple packet losses in a single data window with underutilization of bandwidth. SACK TCP has got information about particular segment which are received successfully. Still after a long span of research in the same field, none of the variants is able to correctly discriminate between the causes of losses and treat them accordingly [1], [2].
3.1.2 Detection and Recovery of Corruption Losses

Link layer detect and discard corrupted frames. However, some corrupted packet may be left undetected and delivered to the point-to-point protocol (PPP) running over the link. PPP protocol is commonly used as a link-layer service for Internet layer. PPP provides check sum of the payload and is capable of detecting the most corrupted frames. Corrupted packet is handed over to the IP layer. The checksum used at IP layer protects only the header and not the datagram’s payload. Routers in the Internet are required to check only IP checksum and not the checksum IP datagram’s payload. Hence, packets with corrupted IP headers are discarded at the first router but the packets with corrupted payload delivered to the TCP.

If the TCP header or payload is corrupted, the packet is transmitted all the way through the Internet to the destination. Apparently, the transmission of corrupted packets through the Internet wastes resources. A checksum used by TCP covers the TCP header, the payload and the pseudo-header composed of IP source and destination addresses and the length of TCP segment. The TCP checksum detects most of the corrupted packets, but still there is some possibility of the corrupted data being delivered to applications. TCP takes no actions upon a packet with an invalid checksum. Such packets are silently discarded. Since TCP silently discards corrupted packets, the recovery procedure is the same whether a packet is delivered to TCP with corruption or lost. As seen previously, three *dupacks* or a retransmission timeout is used for detection of a packet loss. Upon detection, the packet is retransmitted and congestion control action takes place [12], [23], [24].

In wireless channels, TCP packets may frequently be lost due to corruption rather than congestion, and TCP's response of slowing its sending rate for all losses can severely and
unnecessarily degrade performance. Since most applications in common use today (web, email, file transfer) run over TCP, they can perform more poorly than necessary over wireless networks.

SACK TCP’s strength lies in its ability to avoid unnecessary retransmissions based on the SACK information available from receiver. SACK TCP is able to recover faster than New Reno TCP from losses because it is able to avoid retransmission of the packets that have certainly reached the receiver. Avoiding unnecessary retransmission allows SACK TCP to utilize the available bandwidth more efficiently, which results in overall improvement of protocol with reference to not only throughput, but also fair sharing of bandwidth of competing traffics. SACK information provided by a TCP receiver can be used more effectively, considering the fact that receiver selectively acknowledges the recent packet it has received through outgoing acknowledgements. Further a receiver would need to acknowledge a packet selectively in case of an intermediate loss from the last acknowledgement in order. This information arriving at sender can be useful to estimate network’s condition [25].

3.2 Approaches to Improve Wireless TCP Performance

Traditional TCP schemes may suffer from severe performance degradation in a mixed wired and wireless environment. Modifications to standard TCP to remedy its deficiency in wireless communications have been a vigorous area of research. Many schemes have been proposed for TCP based on applications for different wireless applications. There are various wireless environments and hence wireless TCP can be precisely designed to satisfy their needs. The
most common wireless networks include satellite networks, ad hoc networks, and general wireless platforms for instance wireless LANs and cellular systems. The design of wireless TCP considers the characteristics of a particular type of wireless network and its needs; for instance, a satellite network has long propagation delay whereas an ad hoc network is infrastructure less. Nevertheless, an obstacle that all wireless networks have to face is high BERs. In heterogeneous networks, to explicitly differentiate the cause of packet loss becomes the foremost goal of TCP design. Such efforts aim to find an explicit way to inform the sender of the cause of packet dropping i.e. congestion or random errors. Therefore, the sender is able to make appropriate decisions for \( cwnd \) adjustment. The standard Reno scheme halves its window size when facing a packet loss irrespective of the cause. If the loss is due to network congestion, this behavior alleviates network congestion. However, it would degrade the performance in case of loss due to random loss [26].

### 3.2.1 Satellite Networks

TCP schemes for satellite networks are designed based on the observation that it takes the sender of a TCP connection a fairly long time to reach a high sending rate throughout than the traditional slow start phase. While many TCP applications such as HTTP are based on the transfer of small files, with the long propagation delay of the satellite link it can happen that the entire transfer occurs within the slow start phase and the connection is not capable of completely utilizing the available network bandwidth.

- **TCP Peach**

TCP Peach employs two new mechanisms, sudden start and rapid recovery, in combination with the traditional TCP’s congestion avoidance and fast retransmit algorithms [27]. It is
done to cope up with TCP performance degradation over satellite communication links characterized by large propagation delay and high link error rates. Sudden start introduces the use of dummy packets that are copies of the last packet of data sent by the sender. The sender sends multiple dummy packets between two consecutive data packets; the reception of ACKs for the dummy packets indicates the availability of the unused network resource, and the sender is triggered to increase its sending window quickly [28]. Dummy packets are labeled with low priority; and hence they are dropped first by a router as the congestion occurs. Simulations show that sudden start can reach the available bandwidth within two RTTs, whereas traditional slow start in TCP-Reno takes about seven RTTs to reach at the same bit error rate. Rapid recovery replaces the classical fast recovery method in order to enhance throughput in the occurrence of high link error rates. In the rapid recovery phase, instead of trying to distinguish congestive loss from transmission loss, the sender uses the lower-priority dummy packets to interleave the data packets and inflates the sending window size upon reception of ACKs for the dummy packets [29], [30], [31].

3.2.2 Ad Hoc Networks

In an ad hoc network, high bit errors rates, frequent route changes and network partitions are typical [32]. These characteristics result in packet loss besides network congestion, and therefore should be treated in a different way.

- ATCP

ATCP is designed as an end on solution to enhance TCP throughput for such an environment. It is realized as a thin layer introduced between the standard TCP and IP layers. It relies on Explicit Congestion Notification (ECN) to detect congestion and distinguish congestion loss.
from error loss. ICMP destination unreachable message is used to detect an alteration of route or transitory partition in ad hoc network. [33]. According to these feedbacks from the network, the ATCP layer puts the TCP sender into either retransmit, persist, congestion control state accordingly. When the packet loss is due to a high BERs, ATCP retransmits the lost packet so that TCP does not perform congestion control; if the ICMP message indicates a route change or network partition, ATCP places the TCP sender into persist state waiting for router’s reconnection. For packet reallocation, ATCP rearrange the packets so that TCP would not generate dupacks. TCP enters the normal congestion control stage on indication of real congestion by ECN. By inserting the ATCP layer between the TCP and IP layers, the scheme does not modify the TCP code; in addition, since the ATCP layer does not generate or regenerate ACK packets by itself, the end-to-end TCP semantic is maintained [34], [35].

### 3.2.3 Cellular Network

Cellular networks, where the base station interconnects a fast fixed network and a slow mobile network, modifications of TCP algorithms focus on cellular characteristics such as handoff and the commonly shared problem of all wireless networks, high BERs.

- **Freeze TCP**

Freeze TCP is an end-to-end solution to improve TCP performance in the mobile environment. It imposes no restrictions on routers and only necessitate code alterations at the mobile unit or receiver side. It addresses the throughput degradation caused by frequent disconnections (and reconnections) due to mobile handoff or momentary blockage of radio signal due to obstacles. The assumption is that the mobile unit has knowledge of the radio signal strength, and thus can predict the impending disconnections. The Freeze TCP receiver
on the mobile unit proactively sets the advertised window size to zero in the ACK packets in the occurrence of impending disconnection. Because the TCP sender selects the window size to be the minimum of its vision of the window size and the receiver’s advertised window size, this zero window size ACK packet would force the sender into persist mode, where it blocks transmitting more packets, keeping its sending window unchanged. To prevent the sender from exponentially backing off, when it detects the reconnection the receiver sends several positive ACK packets to the sender acknowledging the last received packet before disconnection so that the transfer can resume quickly at the rate before the disconnection occurs. To implement the scheme proposed in, cross-layer information must be exchanged, and the TCP layer protocol must be exposed to some details of the roaming and handoff algorithms implemented by Network Interface Card (NIC) manufacturers on the interface devices [1].

From the implementation point of view, wireless TCP algorithms can be planned in either of the two modes viz. split mode or end to end. Due to the significant difference in characteristics between wireless and wired links, the split mode discussed in next section divides the TCP connection into wireless and wired portions, and ACKs are generated for both portions separately. By doing so, the performance on the wired portion is least affected by the relatively unreliable wireless portion. On the other hand, the end to end mode discussed in subsequent section treats the route from the sender to the receiver as an end to end path, and the sender is acknowledged directly by the receiver. This maintains the end to end semantics of the baseline TCP design.
3.3 Existing Schemes Available

3.3.1 Split Mode

The wired portion of the heterogeneous network is more reliable than the wireless portion in terms of link capacity and bit error rates. However, transmission would be holdup at a slow and lossy wireless link. Split mode attempts to shield the wireless portion from the fixed network by separating the flow control at the intermediate router (or a base station for a cellular network), such that the behavior has the slightest impact on the fixed network. The intermediate router behaves as a terminal in case of both the wired and wireless portions. Both end hosts communicate with the intermediate router autonomously without acquaintance of the other end. The intermediate router is reinforced with functionalities to manage the transaction among the two network portions [36]. This is illustrated in Figure 3.3.1. In Indirect TCP (I-TCP), the Mobile Support Router (MSR) connects the Mobile Host (MH) to the Fixed Host (FH), and establishes two separate TCP connections with the FH and MH, respectively. The MSR communicates with the FH on behalf of MH. The congestion window is preserved discretely for the wireless and fixed connections. A new MSR takes over the communication with FH impeccably, as soon as the MH switches the cell. Thus, the FH is protected from unpredictable feature of wireless connections. [36], [37].

![Figure 3.3.1: Split Connection of wireless TCP](image)

Figure 3.3.1: Split Connection of wireless TCP
3.3.2 End to End Approach

In split mode, an intermediate router has to reveal the information in the TCP packet and procedure associated data before it reaches the MH respectively. The MSR communicates with the FH in aid of the MH whereas in the end to end approach, just the end hosts participate in flow control. The receiver provides feedback reflecting the network condition, and the sender makes choices for congestion control. This is illustrated in Figure 3.3.2. In the end to end approach, the ability to accurately probe for the available bandwidth is the key to better performance, which is nevertheless a great challenge. The available bandwidth of a flow is the minimum idle link capacity of the flow’s reasonable share along the path. [38].

The end to end approach can have its congestion control mechanism realized in two ways, reactive and proactive. By reactive congestion control, the sender rectifies the congestion window when the network condition becomes marginal or else crossed a threshold. By proactive congestion control, feedbacks from the network guide the sender to reallocate network resources so as to prevent congestion.

![Figure 3.3.2: End-to-End Connection of wireless TCP](image)

- **Reactive Congestion Control**: The standard Reno scheme utilizes reactive flow control. The congestion window is adjusted based on the collective feedback of ACKs and dupacks generated on the receiver. Many TCP schemes have been suggested as an improvement to the standard Reno scheme in this reactive manner. A single packet drop inside one window is
taken care by the fast retransmit algorithm of Reno. Reno terminates the fast recovery mechanism after the recovery of one lost packet. The Reno scheme would be forced to invoke multiple fast recovery procedures back and forth, slowing down the retrieval of the lost packet. New Reno modifies the fast recovery mechanism of Reno to cope with multiple losses from a single window. In New Reno the fast recovery mechanism does not terminate until multiple losses, informed by the reaction of partial ACKs, from single window are all recovered. The limitation of New Reno is, however, that it cannot distinguish the cause of the packet loss. TCP SACK has a selective acknowledgement option for TCP, targeting the similar problem which Reno tries to handle. While the feedback of Reno and New Reno is dependent on cumulative ACKs, SACK utilizes a selective repeat retransmission policy. It indicates a block of data that has been successfully received and queued at the receiver when packet loss occurs rather than sending a partial ACK as in New Reno. These TCP variants are discussed in more detail in Section 2.2.

- **Proactive Congestion Control:** In proactive congestion control, the sender attempts to adjust the congestion window proactively to an optimal rate according to the information collected via feedback, which may be transformed to an indication of the network status. By doing so, the sender reacts intelligently to the network condition or the cause of the packet loss, due to either congestion or random BERs, and therefore prevents the network from entering an undesired state (e.g., congestion or pointless reduction of the congestion window).

Different strategies can be utilized in the design to provide the sender with the explicit network condition. TCP Vegas analyzes the backlogged packets in the buffer of the holdup
link. Vegas selects the minimal RTT as a reference to derive the optimal throughput the network can accommodate. It also records the real transmission rate at the sender throughout transmission to gain the authentic throughput. The difference between the optimal throughput and the actual sending rate can be used to derive an amount of mangled data in the network. It then sets two thresholds corresponding to two network stages, one of which represents too little backlogged data, the other too much. For the former stage, Vegas increase the congestion window linearly; for the latter, Vegas decrease the congestion window linearly. In doing so, Vegas try to stabilize the network congestion state around the optimal point by proactively adjusting the congestion window without affecting it much.

TCP Veno adopts the same procedure as that of Vegas to obtain the backlogged packets in the network. It further proposes a way to distinguish the cause of packet loss. If the number of backlogged packets is less than threshold, the loss is considered to be arbitrary, otherwise it is considered as congestive. If the loss is due to congestion, Veno practices the standard Reno scheme. For loss due to a random error, it increases the congestion window in a conservative manner (i.e., transmitting single packet for each other ACK received).

TCP Westwood is a rate-based end-to-end approach in which the sender estimates the available network bandwidth dynamically by measuring and averaging the rate of returning ACKs. Under the assumption of an ideal error-free and congestion-free reverse path, the inter-ACK gap replicates the accessible network resource. Westwood deploys an available bandwidth measurement module at the sender side based on the period of inveterate ACKs. It calculates the explicit available bandwidth and using it guides the sending rate. When TCP Westwood determines that the link is congested after having received three dupacks, it sets
the slow start threshold to replicate its estimated bandwidth-delay product. TCP-Westwood claims improved performance over TCP Reno and SACK while attaining equality and friendliness. The end to end approach maintains the network layer structure and needs minimum modification at end hosts, and in some cases the routers.

TCP Jersey is another proactive scheme that adapts the sending rate proactively according to the network condition. It consists of two key components, the Congestion Warning (CW) and Available Bandwidth Estimation (ABE) algorithm router configuration. ABE is a TCP sender side addition that continuously estimates the bandwidth available to the connection and guides the sender to adjust its transmission rate when the network becomes congested. CW is a configuration of network routers such that routers alert end stations by highlighting all packets when there is an indication of incipient congestion. The marking of packets by CW configured routers helps the sender of the TCP connection to effectively differentiate packet losses caused by network congestion from the one caused due to wireless link errors. Based on the congestion indication implied by CW and the estimation from ABE, TCP Jersey calculates the optimal congestion window size at the sender. ECN needs to be supported at the routers so as to implement CW. Proactive schemes, in general, exhibit the ability to handle random loss more effectively. The experimental network is a reliable fixed link coupled to a lossy wireless link. The link error rate on the wireless link is varied to show how proactive schemes act upon in the wireless environment compared to reactive schemes. The basic trend is that the performance of both schemes degrades with growing wireless link error rate. However, TCP Jersey, which represents the proactive approach, outperforms the standard TCP Reno considerably in a higher error rate environment. For instance, at 2% error rate, 280 percent more goodput is attained by TCP Jersey than by Reno. This is mainly
due to the fact that proactive schemes are able to better distinguish random loss from congestive loss than reactive schemes. Hence, there are fewer unnecessary decreases of the congestion window in transmission [1], [2], [12], [23], [24], [34], [35].

3.4 Comparison

Comparative summary is tabulated in Table 3.4.1 which evaluates various approaches in different wireless conditions [1].

Table 3.4.1: Comparison of Various Approaches

<table>
<thead>
<tr>
<th>Application specific approach</th>
<th>Schemes</th>
<th>Need intermediary</th>
<th>TCP semantics</th>
<th>Support for mobility</th>
<th>Modification requirement</th>
<th>Targeted application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proactive approach</td>
<td>I-TCP</td>
<td>Yes</td>
<td>Split</td>
<td>High</td>
<td>Base station</td>
<td>Cellular</td>
</tr>
<tr>
<td>TCP-Peach</td>
<td>Yes</td>
<td>End to End</td>
<td>High</td>
<td>Router and end stations</td>
<td>Satellite</td>
<td></td>
</tr>
<tr>
<td>ATCP</td>
<td>No</td>
<td>End to End</td>
<td>High</td>
<td>TCP stack</td>
<td>Ad hoc</td>
<td></td>
</tr>
<tr>
<td>Freeze TCP</td>
<td>No</td>
<td>End to End</td>
<td>High</td>
<td>Base station</td>
<td>Cellular</td>
<td></td>
</tr>
<tr>
<td>Reactive approach</td>
<td>TCP-New Reno</td>
<td>No</td>
<td>End to End</td>
<td>Low</td>
<td>Sender side</td>
<td>Heterogeneous</td>
</tr>
<tr>
<td>TCP-SACK</td>
<td>No</td>
<td>End to End</td>
<td>Low</td>
<td>Sender side</td>
<td>Heterogeneous</td>
<td></td>
</tr>
<tr>
<td>TCP-Vegas</td>
<td>No</td>
<td>End to End</td>
<td>Low</td>
<td>Sender side</td>
<td>Heterogeneous</td>
<td></td>
</tr>
<tr>
<td>TCP-Veno</td>
<td>No</td>
<td>End to End</td>
<td>Low</td>
<td>Sender side</td>
<td>Heterogeneous</td>
<td></td>
</tr>
<tr>
<td>TCP-Westwood</td>
<td>No</td>
<td>End to End</td>
<td>High</td>
<td>Sender side</td>
<td>Heterogeneous</td>
<td></td>
</tr>
<tr>
<td>TCP-Jersey</td>
<td>No</td>
<td>End to End</td>
<td>High</td>
<td>Router and sender side</td>
<td>Heterogeneous</td>
<td></td>
</tr>
</tbody>
</table>