CHAPTER 2

LITERATURE REVIEW

2.1 BRIEF SURVEY OF TRANSMISSION CONTROL PROTOCOL

TCP-F (TCP-Feedback) (Chandran et al 1998) relies on the network layer at intermediate hosts to detect the route failures due to the mobility of downstream neighbors along the route. A sender can be in an active state or a snooze state. In the active state, transport is controlled by the normal TCP. As soon as an intermediate host detects a link failure, it explicitly sends a route failure notification (RFN) packet to the sender and records this event.

After receiving the RFN, the sender goes into the snooze state by stopping the moment of further packets and freezing the values of state variables such as retransmission timer and congestion window size. The sender remains in the snooze state until it is notified of the restoration of the route through a route reestablishment notification (RRN) packet from an intermediate host. Then it enters the active state again.

Explicit Link Failure Notification (ELFN) (Holland and Vaidya 1999) is another technique based on feedback. The objective is to provide the TCP sender with information about link and route failures so that it can avoid responding to the failures as if congestion happened. ELFN is based upon
DSR routing protocol. To implement ELFN message, the route failure message of DSR is modified to carry a payload similar to the ‘host unreachable’ ICMP message.

When a TCP sender receives an ELFN, it disables its retransmission timers and enters a stand-by" mode, which is similar to the snooze state of TCP-F. Instead of using an explicit notice to signal that a route has been reestablished, a packet is sent periodically to probe the network to see if a route has been established. After finding a new route, the sender leaves the stand-by mode, restores its retransmission timers and continues as normal.

The fixed RTO technique (Balakrishnan et al 1996) does not rely on the feed-back from lower layers. In fact, a heuristic was employed to distinguish route failures and congestion. When timeouts occur consecutively, the sender assumes that a route failure has happened rather than network congestion. The unacknowledged packet is retransmitted again but the RTO is not doubled a second time. The RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged. This technique complements the out-of-order detection and response technique.

A TCP (Liu and Singh, 2001) does not impose changes to the standard TCP itself. Rather it implements an intermediate layer between network and transport layers in order to lead TCP to an enhanced performance and still maintains inter operation with non-ATCP machines. In particular, this approach relies on the Internet Control message protocol (ICMP) and Explicit Congestion Notification (ECN) scheme to detect network partition and congestion, respectively. In this way, the intermediate layer keeps track of
the packets to and from the transport layer so that the TCP congestion control is not invoked when it is not really needed, which is done as follows.

When three duplicate ACKs are detected, indicating a lossy channel, a TCP puts TCP in “persists mode” and quickly retransmits the lost packet from the TCP buffer; after receiving the next ACK the normal state is resumed. In case an ICMP “Destination Unreachable” message arrives, pointing out a network partition, A TCP also puts the TCP in “persists mode” which only ends when the connection is reestablished. At last, when network congestion is detected by the receipt of an ECN message, the A TCP does nothing but forwards the packet to TCP so that it can invoke its congestion control normally.

This model was implemented in a test bed and evaluated under different constraints such as congestion, lossy scenario, partition, and packet reordering. In all cases the transfer time of a given file by A TCP yielded better performance comparatively to TCP. However, again the used scenario was somewhat special, since neither wireless links nor ad hoc routing protocols were considered.

In fact, such experiments relied on simple Ethernet networks connected in series in which each node had two Ethernet cards. More over, some assumptions such as ECN-capable nodes as well as sender node being always reachable might be somehow hard to be met. In case the latter is not fulfilled, for example, the ICMP message might not even reach the sender which would retransmit continuously instead of entering “persists mode”.
Also, ECN scheme deployment raises security concerns [ECN], and it might compromise the viability of this scheme. In summary, as shown by the simulations, these feedback-based approaches improve TCP performance significantly while maintaining TCP’s congestion control behavior and end-to-end TCP semantics.

However, all these schemes require that the intermediate nodes have the capability of detecting and reporting network states such as link breakages and congestion. Enhancement at the transport layer, network layer, and link layer are all required. It deserves further research on the ways to detect and distinguish network states in the intermediate nodes.

TCP-Bus (Dongkyun Kim et al 2000) is similar to TCP-F in detection mechanisms. Two control messages (ERDN and ERSN) related to route maintenance are introduced to notify the TCP sender of route failures and route reestablishment. These indicators are used to differentiate between network congestion and route failures as a result of node movement. ERDN (Explicit Route Disconnection Notification) message is generated at an intermediate Node upon detection of a route disconnection, and is propagated toward the sender.

After receiving an ERDN message, the sender stops transmission. Similarly, after discovering a new partial path from the failed node to the destination, the failed node returns an ERSN (Explicit Route Successful Notification) message back to the sender. On receiving ERSN Message, the sender resumes data transmission. TCP-Bus considers the problem of reliable transmission of control messages.
If a node A reliably sends an ERDN message to its upstream node B, the ERDN message subsequently forwarded by node B can be overheard by A (assuming same transmission ranges of A and B). Thus, if a node has sent an ERDN message but cannot overhear any ERDN message relayed by its upstream node during a certain period, it concludes the ERDN message is lost and retransmits it. The reliable transmission of ERSN is similar.

To summarize, all these mechanisms rely on the intermediate nodes, where the route failures are detected, to send some control messages to notify the TCP sender. They are categorized and called the network layer feedback mechanisms.

A lot of research regarding TCP over wireless links has been undertaken. One much cited work is (Maleki et al 2003), which studies TCP performance related to the 802.11 MAC layer. This simulation study shows that the relation between MAC and TCP causes severe unfairness conditions between TCP flows. Both the 802.11 MAC layer and TCP have back-off timers.

These timers affect each other to give a very low throughput for TCP when there are packet collisions due to the hidden terminal problems. This work also demonstrates conflicts between TCP data packets and TCP ACKs going in different direction in a wireless network when the window sizes are greater than one.

Some related problems are the focus in (Xu and Saadawi 2001). This work demonstrates instability problems of TCP connections over wireless links. The work is based upon simulations in ns-2, and shows that the TCP maximum window size has an effect on the problem. Like (Gerla et al
1999), it also shows that there are problems between the 802.11 MAC and TCP, and that this gives unfairness between TCP flows.

Another paper that investigates TCP problems for mobile ad hoc networks is (Fu et al 2000). Unlike the previous work, this paper tries to model a realistic scenario, by doing simulations with ns-2. This includes better models for mobility, channel error, and shared-channel contention. Their results show that network disconnections and reconnections due to mobility have the most significant impact on TCP’s performance in mobile ad hoc networks.

A recent paper (Fu et al 2003) shows that TCP’s throughput and loss can be minimized for a given network topology and node mobility, with an optimized TCP window size. TCP achieves best throughput via improved spatial channel reuse.

Several papers try to improve TCP’s performance over wireless links. Most of these are related to wireless cells, where there are either a base satellite station or an 802.11 access point. Recently, there has also been focus on the TCP problem for wireless, multi-hop, ad hoc networks. The study (Andreas Hafslund 2004 and Elaarag 2002) gives an overview of the TCP problem, and also a survey of possible solutions to the problem.

Sack (Mohsin Ali et al 2011) is a type of selective acknowledgements for the TCP to provide the sender with sufficient information to recover quickly from multiple packet losses within a single transmission window (Ahmed 2004 and Chrungoo 2001) and (Fall and Sally Floyd 1996). Each acknowledgement contains information about up to three noncontiguous blocks of data that have been received successfully by the
receiver. Each block of data is described by its starting and ending sequence numbers describing the left and right edges of blocks of received data. The congestion control actions are performed at the sender whenever losses occur.

*Sack* uses *Reno’s Fast Recovery* algorithm and each packet loss leads to congestion avoidance as compared to new reno (once for all in single window). TCP with *Sack* option performs better than standard TCP in situations where there are multiple packet losses within a window of outstanding data (Mathis et al 1996). However, this scheme is not good when the sender’s window size is small. Moreover, there is another approach called delayed acknowledgement which is discussed in (Haifa Touati et al 2007).

Many researches have since focused on mechanisms to enhance TCP performance in wireless environments. Examples of such schemes include TCP Westwood (TCPW). TCP Westwood has been initially designed in [20]. This protocol relies on a simple modification of the TCP source protocol behaviour for a faster recovery. This is performed by setting both a slow start threshold and a congestion window values that result from the effective connection while congestion is experienced. Hence, TCPW attempts to make a more “informed” decision, in contrast with TCP Reno, which automatically halves the congestion window after three duplicate ACKs. Like TCP Reno, TCPW cannot distinguish between buffer overflow losses and random losses.

However, in the presence of random losses, TCP Reno overreacts and reduces the window by half. Whereas, after a packet loss, TCPW resumes with the previous window as long as the bottleneck is not yet saturated (i.e. there is no buffer overflow) (Gerla et al 2001, Wang et al 2002). The TCPW ABSE (Adaptive Bandwidth Share Estimation) protocol has been proposed in
(Casetti et al 2002). It palliates TCP efficiency degradation in packet loss environment. TCPW+, described and studied in (Ferorelli et al 2002), is intended to improve TCPW performance in the case of Internet transmissions. TCP Westwood+ is a slightly modified version of the bandwidth estimation algorithm proposed in (Grieco and Mascolo 2002) to cope up with ACK compression effect.

Nevertheless, it is noted via a series of simulations that TCP New Reno, TCPW ABSE and TCP Westwood+ throughput drop significantly in presence of continuous nodes’ mobility. To address this issue, a new protocol is proposed TCP AR (Adaptive RTO) (Haifa Touati et al 2007) protocol to enhance TCP performance in mobile environments by adapting RTO’s (Retransmission Time-Out) value to network conditions, while preserving both the TCP New Reno principle and TCPW ABSE throughput estimation.