International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.3, Issue.1, Jan-Feb. 2013 pp-439-446 ISSN: 2249-6645

TCP ISSUES IN MOBILE ADHOC NETWORKS: Challenges and Solutions

Jatau Isaac Katuka, [1] Yahaya Shagaiya Daniel, [2] Bako Sunday Samuel[3]
[123] *Department of Mathematical Sciences, Kaduna State University. Nigeria.*

Abstract: *The transmission control protocol (TCP) is the most predominant transport layer protocol in the Internet today. It transports more than 90% percent of the traffic on the Internet. Its reliability, end-to-end congestion control mechanism, byte-stream transport mechanism, and, above all, its elegant and simple design have not only contributed to the success of the Internet, but also have made TCP an influencing protocol in the design of many of the other protocols and applications. Its adaptability to the congestion in the network has been an important feature leading to graceful degradation of the services offered by the network at times of extreme congestion. TCP in its traditional form was designed and optimized only for wired networks. This protocol is mainly known because of its feature of providing reliable delivery of packets from source to destination. In TCP, reliability is achieved by retransmitting lost packets. Since TCP is widely used today and the efficient integration of mobile adhoc networks (MANETs) with the Internet is paramount wherever possible, it is essential to have mechanisms that can improve TCP's performance in MANETs. In this paper, we describe the challenges of standard TCP in MANETs. We also proposed some techniques to improve TCP performance, advantages and disadvantages of the proposed techniques.*

Keyword: *Mobile Adhoc Networks (MANETs), Mobility, TCP Feedback, Medium Access Control (MAC), Transmission Control Protocol (TCP), Wireless Networks.*

I. Introduction

The wireless arena has been experiencing exponential growth. We have seen great advances in network infrastructures growing availability of wireless applications, and the emergence of omnipresent wireless devices such as portable or handheld computers, Personal Digital Assistant *(PDA)*, and all cell phones, all getting more powerful in their capabilities. These devices are now playing an ever-increasing important role in our lives. To mention only a few examples, mobile users can rely on their cellular phones to check e-mail and browse the internet; travellers with portable computers can surf the internet from airports, railway stations, cafes, and other public locations; tourists can use GPS terminals installed inside rental cars to view driving maps and locate tourist attraction; and at home, a family can synchronize data and transfer files between portable devices and desktops. Wireless internetworks are inevitable in future. Wireless links have high bit error rate, high latencies, and low bandwidth as compared to a wired link. The bit error is high because of its vulnerability to interference by Gaussian and stray noise and signals. *TCP* on wireless triggers congestion avoidance mechanisms, wherein the congestion window is reduced exponentially, thereby reducing the effective window size. The exponential decrease is to avoid further packet loss at the router. *TCP* is a reliable, end-to-end, connection-oriented transport layer protocol that provides a byte-stream-based service [the stream of bytes from the application layer is split into *TCP* segments, the length of each segment limited by a maximum segment size (*MSS*)]. The major responsibilities of *TCP* include congestion control, flow control, in-order delivery of packets, and reliable transportation of packets. Congestion control deals with excess traffic in the network which may lead to degradation in the performance of the network, whereas flow control controls the per-flow traffic such that the receiver capacity is not exceeded. *TCP* regulates the number of packets sent to the network by expanding and shrinking the congestion window. The *TCP* sender starts the session with a congestion window value of one *MSS*. It sends out one *MSS* and waits for the ACK. Once the ACK is received within the retransmission timeout (*RTO*) period, the congestion window is doubled and two MSSs are originated. This doubling of the congestion window with every successful acknowledgment of all the segments in the current congestion window, is called *slow-start* (a more appropriate name would be *exponential start*, as it actually grows exponentially) and it continues until the congestion window reaches the *slow-start threshold*. This linear growth, which continues until the congestion window reaches the receiver window (which is advertised by the *TCP* receiver and carries the information about the receiver's buffer size), is called *congestion avoidance*, as it tries to avoid increasing the congestion window exponentially, which will surely worsen the congestion in the network. *TCP* updates the *RTO* period with the current round-trip delay calculated on the arrival of every ACK packet. If the ACK packet does not arrive within the *RTO* period, then it assumes that the packet is lost. *TCP* assumes that the packet loss is due to the congestion in the network and it invokes the congestion control mechanism. The *TCP* sender reduces the slow-start threshold to half the current congestion window or two MSSs whichever is larger, resets the congestion window size to one *MSS*, activates the slow-start algorithm, and resets the *RTO* with an exponential back-off value which doubles with every subsequent retransmission. The slow-start process further doubles the congestion window with every successfully acknowledged window and, upon reaching the slow-start threshold, it enters into the congestion avoidance phase. The *TCP* sender also assumes a packet loss if it receives three consecutive duplicate ACKs (DUPACKs). Upon reception of three DUPACKs, the *TCP* sender retransmits the oldest unacknowledged segment. This is called the *fast retransmit* scheme. When the *TCP* receiver receives out-of-order packets, it generates DUPACKs to indicate to the *TCP* sender about the sequence number of the last in-order segment received successfully.

II. MANETs Application

Because MANETs are flexible networks that can be setup anywhere at any time without infrastructure, including pre-configuration or administration, people have come to realize the commercial potential and advantages that MANETs can bring. Historically, MANETs have primarily been used for tactical network-related applications to improve battle field communications and survivability. The dynamic nature of military operations means it is not possible to rely on access to a fixed preplaced communication infrastructure on the battle field. Pure wireless communication also has the limitation that radio signals are subject to interference and radio frequencies higher than 100MHz rarely propagate beyond Line of Sight *(LOS)*. MANETs creates a suitable framework to address these issues, provides a mobile wireless distributed multihop wireless network without preplaced infrastructure, and provide connectivity beyond LOS. Although early MANETs applications and deployments were military oriented, non-military applications have grown substantially since then and have become the main focus of today. The introduction of technologies such as Bluetooth [8], IEEE 802.11[12], and Hyperlan[10] greatly facilitate the deployment of adhoc technology outside the military domain.

Applications	Description/Services
Tactical Networks	Military communication, operations Automated Battle fields
Sensor Networks	Collection of embedded sensor devices used to collect real-time data to automate everyday functions. Data highly correlated in time and space, e.g. remote sensors for weather, earth activities etc.
Emergency services	Search-and-rescue operations as well as disaster recovery e.g. early retrieval and transmission of patient data from/to the hospital
Commercial environments	E-Commerce e.g. electronic payments from anywhere
Home and enterprise networking	Home/office wireless networking (WLAN) e.g. shared whiteboard application, use PDA to print anywhere etc.
Educational applications	Setup virtual classrooms or conference rooms, setup adhoc communication during conferences, meetings, or lectures
Entertainment	Multiuser games, robotic pets, etc.
Location-aware services	Follow-on services, e.g. automatic call forwarding, transmission of the actual workspace to the current location.

Table 1. Mobile Adhoc Network Applications

III. TCP's challenges in MANETs

The performance of *TCP* degrades in MANETs, because *TCP* has to face challenges due to several reasons. Most of the progress made in *TCP* is centered on error recovery and congestion control. Representative innovations include fast transmissions and recoery[3], selective acknowledgements[6], random early detection(RED, [5]) in routers, and explicit congestion notification (ECN, [4]). *TCP* assumes that network congestion has happened whenever a packet is lost. It then invokes appropriate congestion control actions including window size reduction. Although this assumption is reasonable for wired networks, it is questionable for wireless networks especially MANETs. Below are the challenges faced when deploying *TCP* over MANETs?

1.1 Media Access Control (MAC)

In MANETs, use of broadcasting and shared transmission media introduces a non-negligible probability of packet collisions and media contention. In addition, with half-duplex radio collision detection is not possible, which severely reduces channel utilization as well as throughput, and brings new challenges to conventional CSMA/CD-based and MAC protocols in general. Among the top issues are hidden-terminal and exposed-terminal problems. The hidden-terminal problem occurs when two or more terminals, say A and B cannot detect each other's transmissions due to being outside of each other transmission range but their transmission ranges are not disjoint. A collision may occur when terminal A and B start transmitting towards the same receiver, terminal C as shown in fig.1 below. The exposed-terminal problem results from situations in which a permissible transmission from mobile station (sender) to another station has to be delayed due to the irrelevant transmission range. Fig. 1 below depicts a typical scenario in which the exposed-terminal problem may occur. It is worth noting that hidden-terminal and exposed-terminal problems are correlated with the transmission range. By increasing the transmission range, the hidden-terminal problem occurs less frequently. On the other hand, the exposed-terminal problem becomes more important as the transmission range identifies the area affected by a single transmission.

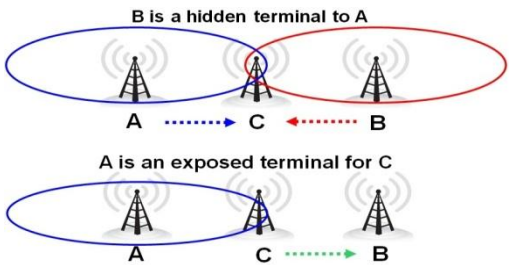

Fig. 1. Hidden-terminal and Exposed-terminal Problems

1.2 Power Constraint: Because batteries carried by each mobile node have limited power, processing power is limited, which in turn limits services and applications that can be supported by each node. This becomes a bigger issue in MANETs because each node is acting as both an end system and a router at the same time, additional energy is required to forward packets from other nodes. *TCP* must use this scarce energy resource in an efficient manner by minimizing the number of unnecessary transmissions at the transport layer as well as at the link layer.

1.3 Mobility: MANETs experience dynamic changes in network topology because of the unrestricted mobility of the nodes in the network. The topology changes lead to frequent changes in the connectivity of wireless links and hence the route to a particular destination may need to be recomputed very often. The responsibility of finding a route and reestablishing it once it gets broken is attached to the network layer. Once a path is broken, the routing protocol initiates a route reestablishment process. This route reestablishment process takes a significant amount of time to obtain a new route to the destination. The route reestablishment time is a function of the number of nodes in the network, transmission ranges of nodes, current topology of the network, bandwidth of the channel, traffic load in the network, and the nature of the routing protocol. If the route reestablishment time is greater than the *RTO* period of the *TCP* sender, then the *TCP* sender assumes congestion in the network, retransmits the lost packets, and initiates the congestion control algorithm. These retransmissions can lead to wastage of bandwidth and battery power. Eventually, when a new route is found, the *TCP* throughput continues to be low for some time, as it has to build up the congestion window since the traditional *TCP* undergoes a slow start.

1.4 Multipath Fading: It is found that the *TCP* throughput degrades rapidly with an increase in path length in string (linear chain) topology MANETs. The possibility of a path break increases with path length. Given that the probability of a link break is p_l, the probability of a path break (p_b) for a path of length k can be obtained as $P_b = 1 — (1 — p_l)^k$. Hence as the path length increases, the probability of a path break increases, resulting in the degradation of the throughput in the network. However, it is doubted by some researcher[15] whether IEEE 802.11 works well in multi-hop adhoc networks, particularly when *TCP* is used as the transport layer protocol. Existing research work [7] has shown that *TCP* performance decreases drastically as the hop count becomes bigger and bigger.

1.5 Misinterpretation of congestion window: *TCP* considers the congestion window as a measure of the rate of transmission that is acceptable to the network and the receiver. In MANETs, the congestion control mechanism are invoked when the network gets partitioned or when a path break occurs. This reduces the congestion window and increases the *RTO* period. When the route is reconfigured, the congestion window may not reflect the transmission rate acceptable to the new route, as the new route may actually accept a much higher transmission rate. Hence, when there are frequent path breaks, the congestion window may not reflect the maximum transmission rate acceptable to the network and the receiver.

1.6 Asymmetric link behavior: The wireless link between a base station and a mobile terminal is asymmetric in nature. Compared with the base station, the mobile terminal has a limited power, processing capability, and buffer space. Asymmetry may manifest in several forms like bandwidth, loss rate asymmetry, and route asymmetry. The radio channel used in MANETs has different properties such as location-dependent contention, environmental effects on propagation, and directional properties leading to asymmetric links. The directional links can result in delivery of a packet to a node, but failure in the delivery of the acknowledgment back to the sender. It is possible for a bidirectional link to become uni-directional for a while. This can also lead to *TCP* invoking the congestion control algorithm and several retransmissions. For example, in multi-rate IEEE 802.11 protocol versions, senders may use the Auto-Rate-Feedback (ARF) algorithm for transmission rates after selection [1].

1.7 Uni-directional path: Standard*TCP* relies on end-to-end ACK for ensuring reliability. Since the ACK packet is very short compared to a data segment, ACKs consume much less bandwidth in wired networks. In MANETs, every *TCP* ACK packet requires *RTS*-*CTS*-Data-ACK exchange in case *IEEE* 802.11 is used as the underlying *MAC* protocol. This can lead to an additional overhead of more than 70 bytes if there are no retransmissions. This can lead to significant bandwidth consumption on the reverse path, which may or may not contend with the forward path. If the reverse path contends with the forward path, it can lead to the reduction in the throughput of the forward path. Some routing protocols select the forward path to be also used as the reverse path, whereas certain other routing protocols may use an entirely different or partially different path for the ACKs. A path break on an entirely different reverse path can affect the performance of the network as much as a path break in the forward path.

1.8 Multipath routing: In wired networks, route failures occur rarely while in MANETs, they are frequent events. The main cause of route failures is node mobility since nodes can move arbitrarily. There exists a set of QoS routing and best-effort routing protocols that use multiple paths between a source-destination pair. There are several advantages in using multipath routing. Some of these advantages include the reduction in route computing time, the high resilience to path breaks, high call acceptance ratio, and better security. For *TCP*, these advantages may add to throughput degradation. These can lead to a significant amount of out-of-order packets, which in turn generates a set of duplicate acknowledgments (DUPACKs) which cause additional power consumption and invocation of congestion control. In

[2], Lundgren et al. have conducted experiments and have subsequently conducted that the origin of this problem is heterogeneous transmission rates, the absence of an acknowledgment for broadcast packets, small packet size of Hello messages, and fluctuations of wireless links.

1.9 Network partitioning and remerging: The randomly moving nodes in MANETs can lead to network partitions. As long as the *TCP* sender, the *TCP* receiver, and all the intermediate nodes in the path between the *TCP* sender and the *TCP* receiver remain in the same partition, the *TCP* connection will remain intact. It is likely that the sender and receiver of the *TCP* session will remain in different partitions and, in certain cases, that only the intermediate nodes are affected by the network partitioning.

1.10 The use of sliding-window-based transmission: *TCP* uses a sliding window for flow control. The transmission of packets is decided by the size of the window, and when the ACKs arrive from a destination, further packets are transmitted. This avoids the use of individual fine-grained timers for transmission of each *TCP* flow. Such a design is preferred in order to improve scalability of the protocol in high-bandwidth networks such as the Internet where millions of *TCP* connections may be established with some heavily loaded servers. The use of a sliding window can also contribute to degraded performance in bandwidth-constrained MANETs where the *MAC* layer protocol may not exhibit short-term and long-term fairness. For example, the popular *MAC* protocols such as *CSMA*/*CA* protocol show short-term unfairness, where a node that has captured the channel has a higher probability of capturing the channel again. This unfairness can lead to a number of *TCP* ACK packets being delivered to the *TCP* sender in succession, leading to a burstiness in traffic due to the subsequent transmission of *TCP* segments.

IV. Proposed Methods to Improve *TCP* over MANETs

The quest to overcome the drawback of *TCP* over MANETs cannot be over emphasized. Lots of research has been done to improve the performance of *TCP* over MANETs and subsequently lots of approaches have been proposed. We categorically classify these approaches as follows:

1.11 TCP Feedback: *TCP Feedback* (*TCP*-F)] [9] proposes modifications to the traditional *TCP* for improving performance in MANETs. It uses a feedback-based approach. *TCP*-F requires the support of a reliable link layer and a routing protocol that can provide feedback to the *TCP* sender about the path breaks. The routing protocol is expected to repair the broken path within a reasonable time period. *TCP*-F aims to minimize the throughput degradation resulting from the frequent path breaks that occur in MANETs. During a *TCP* session, there could be several path breaks resulting in considerable packet loss and path reestablishment delay. Upon detection of packet loss, the sender in a *TCP* session invokes the congestion control algorithm leading to the exponential back-off of retransmission timers and a decrease in congestion window size. This was discussed earlier in this chapter. In *TCP*-F, an intermediate node, upon detection of a path break, originates a route failure notification (*RFN*) packet. This *RFN* packet is routed toward the sender of the *TCP* session. The *TCP* sender's information is expected to be obtained from the *TCP* packets being forwarded by the node. The intermediate node that originates the *RFN* packet is called the failure point (*FP*). The *FP* maintains information about all the RFNs it has originated so far. Every intermediate node that forwards the *RFN* packet understands the route failure, updates its routing table accordingly, and avoids forwarding any more packets on that route. If any of the intermediate nodes that receive *RFN* has an alternate route to the same destination, then it discards the *RFN* packet and uses the alternate path for forwarding further data packets, thus reducing the control overhead involved in the route reconfiguration process. Otherwise, it forwards the *RFN* toward the source node. When a *TCP* sender receives an *RFN* packet, it goes into a state called *snooze.* In the snooze state, a sender stops sending any more packets to the destination, cancels all the timers, freezes its congestion window, freezes the retransmission timer, and sets up a route failure timer. This route failure timer is dependent on the routing protocol, network size, and the network dynamics and is to be taken as the worst-case route reconfiguration time. When the route failure timer expires, the *TCP* sender changes from the snooze state to the *connected* state. As soon as a node receives an *RRN* packet, it transmits all the packets in its buffer, assuming that the network is back to its original state. This can also take care of all the packets that were not acknowledged or lost during transit due to the path break. In fact, such a step avoids going through the slow-start process that would otherwise have occurred immediately after a period of congestion. The route failure timer set after receiving the *RFN* packet ensures that the sender does not remain in the snooze state indefinitely. Once the route failure timer expires, the sender goes back to the connected state in which it reactivates the frozen timers and starts sending the buffered and unacknowledged packets. This can also take care of the loss of the *RRN* packet due to any possible subsequent congestion. *TCP*-F permits the *TCP* congestion control algorithm to be in effect when the sender is not in the snooze state, thus making it sensitive to congestion in the network.

Advantages and Disadvantages

TCP-F provides a simple feedback-based solution to minimize the problems arising out of frequent path breaks in MANETs. At the same time, it also permits the TCP congestion control mechanism to respond to congestion in the network. TCP-F depends on the intermediate nodes' ability to detect route failures and the routing protocols' capability to reestablish a broken path within a reasonably short duration. Also, the FP should be able to obtain the correct path (the path

which the packet traversed) to the TCP-F sender for sending the RFN packet. This is simple with a routing protocol that uses source routing [*i.e.*, dynamic source routing (DSR)]. If a route to the sender is not available at the FP, then additional control packets may need to be generated for routing the RFN packet. TCP-F has an additional state compared to the traditional TCP state machine, and hence its implementation requires modifications to the existing TCP libraries. Another disadvantage of TCP-F is that the congestion window used after a new route is obtained may not reflect the achievable transmission rate acceptable to the network and the TCP-F receiver.

4.2 TCP with Explicit Link Failure Notification

Holland and Vaidya proposed the use of *TCP* with explicit link failure notification (TCP-ELFN) [11] for improving *TCP* performance in ad hoc wireless networks. This is similar to *TCP*-F, except for the handling of explicit link failure notification (ELFN) and the use of *TCP* probe packets for detecting the route reestablishment. The *ELFN* is originated by the node detecting a path break upon detection of a link failure to the *TCP* sender. This can be implemented in two ways: (i) by sending an ICMP destination unreachable (DUR) message to the sender, or (ii) by piggy-backing this information on the RouteError message that is sent to the sender. Once the TCP sender receives the ELFN packet, it disables its retransmission timers and enters a standby state. In this state, it periodically originates probe packets to see if a new route is reestablished. Upon reception of an ACK by the TCP receiver for the probe packets, it leaves the standby state, restores the retransmission timers, and continues to function as normal.

Advantages and Disadvantages

TCP-ELFN improves the TCP performance by decoupling the path break information from the congestion information by the use of ELFN. It is less dependent on the routing protocol and requires only link failure notification about the path break. The disadvantages of TCP-ELFN include the following: (i) when the network is temporarily partitioned, the path failure may last longer and this can lead to the origination of periodic probe packets consuming bandwidth and power and (ii) the congestion window used after a new route is obtained may not reflect the achievable transmission rate acceptable to the network and the TCP receiver.

4.3 TCP-Bus: *TCP* with buffering capability and sequence information (*TCP*-Bus) is similar to the *TCP*-F and *TCP-ELFN* in its use of feedback information from an intermediate node on detection of a path break. But *TCP*-Bus is more dependent on the routing protocol compared to *TCP*-F and *TCP-ELFN*. *TCP*-Bus was proposed, with associativity-based routing (*ABR*) [11] protocol as the routing scheme. Hence, it makes use of some of the special messages such as localized query (*LQ*) and REPLY, defined as part of *ABR* for finding a partial path. These messages are modified to carry *TCP* connection and segment information. Upon detection of a path break, an upstream intermediate node [called pivot node (*PN*)] originates an explicit route disconnection notification (*ERDN*) message. This *ERDN* packet is propagated to the *TCP*-Bus sender and, upon reception of it, the *TCP*-Bus sender stops transmission and freezes all timers and windows as in *TCP*-F. The packets in transit at the intermediate nodes from the *TCP*-BuS sender to the *PN* are buffered until a new partial path from the *PN* to the *TCP*-Bus receiver is obtained by the *PN*. In order to avoid unnecessary retransmissions, the timers for the buffered packets at the *TCP*-Bus sender and at the intermediate nodes up to *PN* use timeout values proportional to the round-trip time (*RTT*). The intermediate nodes between the *TCP*-Bus sender and the *PN* can request the *TCP*-Bus sender to selectively retransmit any of the lost packets. Upon detection of a path break, the downstream node originates a route notification (*RN*) packet to the *TCP*-Bus receiver, which is forwarded by all the downstream nodes in the path. An intermediate node that receives an *RN* packet discards all packets belonging to that flow. The *ERDN* packet is propagated to the *TCP*-Bus sender in a reliable way by using an implicit acknowledgment and retransmission mechanism. The *PN* includes the sequence number of the *TCP* segment belonging to the flow that is currently at the head of its queue in the *ERDN* packet. The *PN* also attempts to find a new partial route to the *TCP*-Bus receiver, and the availability of such a partial path to destination is intimated to the *TCP*-Bus sender through an explicit route successful notification (*ERSN*) packet. *TCP*-Bus utilizes the route reconfiguration mechanism of *ABR* to obtain the partial route to the destination. Due to this, other routing protocols may require changes to support *TCP*-Bus. The *LQ* and REPLY messages are modified to carry *TCP* segment information, including the last successfully received segment at the destination. The *LQ* packet carries the sequence number of the segment at the head of the queue buffered at the *PN* and the REPLY carries the sequence number of the last successful segment the *TCP*-Bus receiver received. This enables the *TCP*-Bus receiver to understand the packets lost in transition and those buffered at the intermediate nodes. This is used to avoid fast retransmission requests usually generated by the *TCP*-Bus receiver when it notices an out-of-order packet delivery. Upon a successful *LQ*-REPLY process to obtain a new route to the *TCP*-Bus receiver, *PN* informs the *TCP*-Bus sender of the new partial path using the *ERSN* packet. When the *TCP*-Bus sender receives an *ERSN* packet, it resumes the data transmission. Since there is a chance for *ERSN* packet loss due to congestion in the network, it needs to be sent reliably. The *TCP*-Bus sender also periodically originates probe packets to check the availability of a path to the destination.

Advantages and Disadvantages

The advantages of *TCP*-Bus include performance improvement and avoidance of fast retransmission due to the use of buffering, sequence numbering, and selective acknowledgment. *TCP*-Bus also takes advantage of the underlying routing protocols, especially the on-demand routing protocols such as *ABR*. The disadvantages of *TCP*-Bus include the

increased dependency on the routing protocol and the buffering at the intermediate nodes. The failure of intermediate nodes that buffer the packets may lead to loss of packets and performance degradation. The dependency of *TCP*-Bus on the routing protocol may degrade its performance with other routing protocols that do not have similar control messages as in *ABR*.

4.4 Ad Hoc *TCP*: Similar to *TCP*-F and *TCP-ELFN*, ad hoc *TCP* (*ATCP*) [14] also uses a network layer feedback mechanism to make the *TCP* sender aware of the status of the network path over which the *TCP* packets are propagated. Based on the feedback information received from the intermediate nodes, the *TCP* sender changes its state to the persist state, congestion control state, or the retransmit state. When an intermediate node finds that the network is partitioned, then the *TCP* sender state is changed to the persist state where it avoids unnecessary retransmissions. When *ATCP* puts *TCP* in the persist state, it sets TCP's congestion window size to one in order to ensure that *TCP* does not continue using the old congestion window value. This forces *TCP* to probe the correct value of the congestion window to be used for the new route. If an intermediate node loses a packet due to error, then the *ATCP* at the *TCP* sender immediately retransmits it without invoking the congestion control algorithm. In order to be compatible with widely deployed *TCP*-based networks, ATCP provides this feature without modifying the traditional *TCP*. *ATCP* is implemented as a thin layer residing between the *IP* and TCPprotocols. The *ATCP* layer essentially makes use of the explicit congestion notification (*ECN*) for maintenance of the states. In summary, *ATCP* tries to perform the activities listed in Table 2.

Event	**Action**
Packet loss due to high *BER*	Retransmits the lost packets without reducing congestion window
Route recomputation delay	Makes the *TCP* sender go to persist state and stop transmission until new route has been found
Transient partitions	Makes the *TCP* sender go to persist state and stop transmission until new route has been found
Out-of-order packet delivery due to multipath routing	Maintains *TCP* sender unaware of this and retransmits the packets from *TCP* buffer
Change in route	Recomputes the congestion window

Table 2. The actions taken by ATCP

Advantages and Disadvantages

Two major advantages of *ATCP* are (i) it maintains the end-to-end semantics of *TCP* and (ii) it is compatible with traditional *TCP*. These advantages permit *ATCP* to work seamlessly with the Internet. In addition, *ATCP* provides a feasible and efficient solution to improve throughput of *TCP* in MANETs. The disadvantages of *ATCP* include (i) the dependency on the network layer protocol to detect the route changes and partitions, which not all routing protocols may implement and (ii) the addition of a thin *ATCP* layer to the *TCP*/*IP* protocol stack that requires changes in the interface functions currently being used.

4.5 Split *TCP:* One of the major issues that affects the performance of *TCP* over MANETs is the degradation of throughput with increasing path length, as discussed early in this chapter. The short connections generally obtain much higher throughput than long connections. This can also lead to unfairness among *TCP* sessions, where one session may obtain much higher throughput than other sessions. This unfairness problem is further worsened by the use of *MAC* protocols such as *IEEE* 802.11, which are found to give a higher throughput for certain link-level sessions, leading to an effect known as *channel capture* effect. This effect leads to certain flows capturing the channel for longer time durations, thereby reducing throughput for other flows. The channel capture effect can also lead to low overall system throughput. Split-*TCP* [16] provides a unique solution to this problem by splitting the transport layer objectives into congestion control and end-to-end reliability. The congestion control is mostly a local phenomenon due to the result of high contention and high traffic load in a local region. In the MANET environment, this demands local solutions. At the same time, reliability is an end-to-end requirement and needs end-to-end acknowledgments. In addition to splitting the congestion control and reliability objectives, split-*TCP* splits a long *TCP* connection into a set of short concatenated *TCP* connections (called segments or zones) with a number of selected intermediate nodes (known as proxy nodes) as terminating points of these short connections

Advantages and Disadvantages

Split-*TCP* has the following advantages: (i) improved throughput, (ii) improved throughput fairness, and (iii) lessened impact of mobility. Throughput improvement is due to the reduction in the effective transmission path length (number of hops in a zone or a path segment). *TCP* throughput degrades with increasing path length. Split-*TCP* has shorter concatenated path segments, each operating at its own transmission rate, and hence the throughput is increased. This also leads to improved throughput fairness in the system. Since in split-*TCP*, the path segment length can be shorter than the end-to-end path length, the effect of mobility on throughput is lessened. The disadvantages of split-*TCP* can be listed as

follows: (i) It requires modifications to *TCP* protocol, (ii) the end-to-end connection handling of traditional *TCP* is violated, and (iii) the failure of proxy nodes can lead to throughput degradation. The traditional *TCP* has end-to-end semantics, where the intermediate nodes do not process *TCP* packets, whereas in split-*TCP*, the intermediate nodes need to process the *TCP* packets and hence, in addition to the loss of end-to-end semantics, certain security schemes that require *IP* payload encryption cannot be used. During frequent path breaks or during frequent node failures, the performance of split-*TCP* may be affected.

V. Conclusion

TCP is a reliable transport protocol tuned to perform well in traditional networks made up of links with low bit-error rates. Networks with higher bit-error rates, such as those with wireless links and mobile hosts, violate many of the assumptions made by TCP, causing degraded end-to-end performance. In this paper, we describe several protocols that improve TCP performance in wireless networks. Table 3.illustrates comparison of the different proposed methods based on mechanisms used for increasing efficiency. Different approaches have been taken to tackle the problems, but the right changes is dependent on where the single or multiple modification needs to be done.

Issue	**TCP-F**	**TCP-ELFN**	**TCP-BuS**	**ATCP**	**Split-TCP**
Packet loss due to *BER* or collision	Same as *TCP*	Same as *TCP*	Same as *TCP*	Retransmits the lost packets without involing congestion control	Same as *TCP*
Path breaks	*RFN* is sent to the *TCP*sender and state changes to snooze	*ELFN* is sent to the *TCP*sender and state changes to standby	*ERDN* is sent to the *TCP* sender, state changes to snooze, *ICMPDUR* is sent to the *TCP* sender, and *ATCP* puts *TCP* into persist state	Same as *TCP*	Same as *TCP*
Out-of-order packets	Same as *TCP*	Same as *TCP*	Out-of-order packets reached after a path recovery are handled	*ATCP* reorders packets and hence *TCP* avoids sending duplicates	Same as *TCP*
Congestion	Same as *TCP*	Same as *TCP*	Explicit messages such as *ICMP* source quench are used	*ECN* is used to notify *TCP* sender. Congestion control is same as *TCP*	Since connection is split, the congestion control is handled within a zone by proxy nodes
Congestion window after path reestablishment	Same as before the path break	Same as before the path break	Same as before the path break	Recomputed for new route	Proxy nodes maintain congestion window and handle congestion
Explicit path break notification	Yes	Yes	Yes	Yes	No
Explicit path reestablishment notification	Yes	No	Yes	No	No
Dependency on routing protocol	Yes	Yes	Yes	Yes	No
End-to-end semantics	Yes	Yes	Yes	Yes	No
Packets buffered at intermediate nodes	No	No	Yes	No	Yes

Table 3.A comparison of TCP solutions for MANETs

References

[1] A. Kamerman and L. Monteban, "Wavelan 11: A High – Performance Wireless LAN for the Unlicensed Band," Bell Labs Tech. J. Summer 1997, pp 118-33.
[2] H. Lungren, E. Nordstro, and C. Tschudin, "Coping with Communication Gray Zones in IEEE 802.11 b-Based Adhoc Networks," Proc. Acm Wksp. Wireless Mobile Multimedia, Atlanta, GA, USA, Sept. 2002, pp. 49-55.
[3] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," RFC 2001, Jan. 1997.
[4] K. Ramakrishnan, S. Floyd and D. Black, "The Addition of Explicit Congestion Notification to IP," RFC 3168, Sept. 2001.
[5] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transaction on Networking, Vol 1, no. 4, Aug. 1993.
[6] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, Oct. 1996.
[7] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multihop Networks," In Proceedings of IEEE WMCSA 1999.
[8] "Bluetooth Special Interest Group,"Web site: http://www.bluetooth.com
[9] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback Based Scheme for Improving TCP Performance in Adhoc Wireless Networks," IEEE Personal Communications, 8(1): 34-39, Feb. 2001.
[10] "Broadband Radio Access Networks (BRAN): High Performance Local Area Network (HYPERLAN) Type 2," Tech. Rep. 101 683 V1.1.1, ETSI.
[11] G. Holland, and N.H. Vaidya, "Analysis of TCP Performance Over Mobile Adhoc Networks," MOBICOM' 99, Seattle, Aug. 1999.
[12] "IEEE 802.11 WLAN Standard," Web site: http://standards.ieee.org/getieee802.11
[13] C. Toh, "Associativity-Based Routing for Adhoc Mobile Networks," J. Wireless Pers. Commun., Vol. 4, no. 2, Mar. 1992.
[14] J. Lui, and S. Singh, "ATCP: TCP for Mobile Adhoc Networks," IEEE JSAC, Vol. 19. No. 7, pp. 1300-1315, Jul. 2001.
[15] S. Xu, and T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Adhoc Networks?," IEEE Communi. Magazine, pp. 130-137, June 2001.
[16] S. Kopparty et al., "Split TCP for Mobile Adhoc Networks," Proc. IEEE GLOBECOM, Taiwan, Nov. 2002