We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

ADHOCTCP: Improving TCP Performance in Ad Hoc Networks

Seyed Mohsen Mirhosseini and Fatemeh Torgheh
Islamic Azad University-HidajBranch, Islamic Azad University-AbharBranch
Iran

1. Introduction

A mobile ad-hoc network (MANET) is a special type of wireless networks. It consists of a collection of mobile nodes that are capable of communicating with each other without help from a fixed infrastructure. The interconnections between nodes are capable of changing on a continual and arbitrary basis. Nodes within each other's radio range communicate directly via wireless links, while those that are far apart use other nodes as relays in a multi-hop routing fashion. The typical applications of MANETs include conferences or meetings, emergency operations such as disaster rescue, and battlefield communications.

Transmission Control Protocol (TCP) [1] is a reliable, connection-oriented, full-duplex, transport protocol widely used in wired networks. TCP's flow and congestion control mechanisms are based upon the assumption that packet loss is an indication of congestion. While this assumption holds in wired networks, it does not hold in the case of mobile wireless networks.

In addition to congestion, a transport protocol in an ad hoc network must handle mobility-induced disconnection and reconnection, route change-induced packet out-of-order delivery for mobile hosts, and error/contention prone wireless transmissions. Reaction to these events might require transport control actions different from congestion control. It might be better to periodically probe the network during disconnection than to back off exponentially [2], and it makes more sense simply to re-transmit a packet lost to random channel error than to multiplicatively decrease the current congestion window [3]. Even if the correct action is executed in response to each type of network event, it is not immediately obvious how to construct an engine that will accurately detect and classify events. Packet loss alone cannot detect and differentiate all these new network events [4].

In this paper, we first describe the necessary network states in an ad hoc network to be identified by TCP and use an end-to-end approach for identification of congestion state in ad hoc network then examine metrics that can be measured end-to-end. Two metrics are devised to detect congestion, IDD (Inter Delay Difference) and STT (Short Term Throughput).The approach we propose in this paper utilizes network layer feedback (from intermediate hops) for identification of disconnection state to put TCP sender into persist mode. Therefore we use from advantage of both end to end measurements and network layer feedback.

The remainder of the chapter is organized as follows: It starts with describing TCP's challenges in MANETs environment in Section 2. Section 3 provides an overview of related

works. The design and implementation of ADHOCTCP are presented in Section 4. Simulations results are given in Sections 5.we conclude the chapter in Section 6.

2. Challenges for TCP in MANETs

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. Other than congestion, possible causes of packet losses in MANETs include, wireless link errors, MAC layer losses due to channel contention, and link breakages due to node mobility. All those causes that are not related to congestion can result in unnecessary congestion control, which will degrade the TCP performance.

Unlike wired networks, some unique characteristics of mobile ad hoc networks seriously deteriorate TCP performance. These characteristics include the unpredictable wireless channels due to fading and interference, the vulnerable shared media access due to random access collision, the hidden terminal problem and the exposed terminal problem, and the frequent route breakages due to node mobility. Undoubtedly, all of these pose great challenges on TCP to provide reliable end-to-end communications in mobile ad hoc networks. From the point of view of network layered architecture, these challenges can be broken down into six categories: lossy channels, hidden and exposed stations, network partitions, path asymmetry, route failures, and Energy Efficiency.

2.1 Lossy channels

Wireless links posses high bit error rates that cannot be ignored. But TCP interprets packet losses caused by bit errors as congestion. As a result, its performance suffers in wireless networks when TCP unnecessarily invokes congestion control, causing reduction in throughput and link utilization.

The main causes of errors in wireless channels are the following:

- Signal attenuation: This is due to a decrease in the intensity of the electromagnetic energy at the receiver (e.g. due to long distance), which leads to low signal-to-noise ratio (SNR).
- Doppler shift: This is due to the relative velocities of the transmitter and the receiver. Doppler shift causes frequency shifts in the arriving signal, thereby complicating the successful reception of the signal.
- Multipath fading: Electromagnetic waves reflecting off objects or diffracting around objects can result in the signal traveling over multiple paths from the transmitter to the receiver. Multipath propagation can lead to fluctuations in the amplitude, phase, and geographical angle of the signal received at a receiver.

In order to increase the success of transmissions, link layer protocols implement Automatic Repeat reQuest (ARQ) or Forward Error Correction (FEC), or both. For example, IEEE 802.11 implements ARQ, so when a transmitter detects an error, it will retransmit the frame; error detection is timer based.

Bluetooth implements both ARQ and FEC on some synchronous and asynchronous connections.

Note that packets transmitted over a fading channel may cause the routing protocol to incorrectly conclude that there is a new one-hop neighbor. This one-hop neighbor could

provide a shorter route to even more distant nodes. Unfortunately, this new shorter route is usually unreliable.

2.2 Hidden and exposed stations

In ad hoc networks, stations may rely on physical carrier-sensing mechanisms to determine an idle channel, such as in the IEEE 802.11 DCF function[5]. Contention-based medium access control (MAC) schemes, such as the IEEE 802.11 MAC protocol, have been widely studied and incorporated into many wireless testbeds and simulation packages for wireless multi-hop ad hoc networks, where the neighboring nodes contend for the shared wireless channel before transmitting. There are three key problems, the hidden terminal problem, the exposed terminal problem, and unfairness[6].

Before explaining these problems, we need to clarify the term "transmission range." The transmission range is the range, with respect to the transmitting station, within which a transmitted packet can be successfully received.

A hidden node is the one that is within the interfering range of the intended receiver but out of the sensing range of the transmitter. The receiver may not correctly receive the intended packet due to collision from the hidden node. As shown in Fig. 1, a collision may occur, for example, when terminal A and C start transmitting toward the same receiver, terminal B in the figure. A typical hidden terminal situation is depicted in Fig. 1. Stations A and C have a frame to transmit to station B. Station A cannot detect C's transmission because it is outside the transmission range of C. Station C (resp. A) is therefore "hidden" to station A (resp. C). Since the transmission areas of A and C are not disjoint, there will be packet collisions at B. These collisions make the transmission from A and C toward B problematic. To alleviate the hidden station problem, virtual carrier sensing has been introduced. It is based on a two-way handshaking that precedes data transmission. Specifically, the source station transmits a short control frame, called Request-To-Send (RTS), to the destination station. Upon receiving the RTS frame, the destination station replies by a Clear-To- Send (CTS) frame, indicating that it is ready to receive the data frame. Both RTS and CTS frames contain the total duration of the data transmission. All stations receiving either RTS or CTS will keep silent during the data transmission period (e.g. station C in Fig. 1).

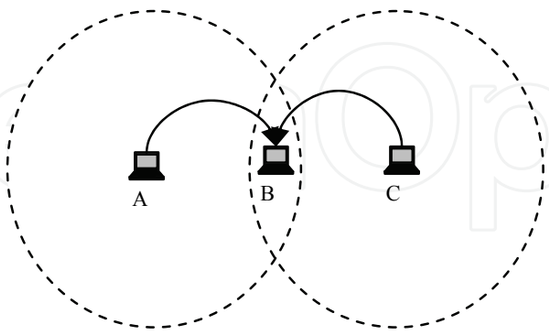


Fig. 1. Hidden terminal problem

However, as pointed out in, the hidden station problem may persist in IEEE 802.11 ad hoc networks even with the use of the RTS/CTS handshake, because the power needed to interrupt a packet reception is much lower than that required to deliver a packet successfully[7,8]. In other words, a node's transmission range is smaller than the sensing node range.

An exposed node is the one that is within the sensing range of the transmitter but out of the interfering range of the receiver. Though its transmission does not interfere with the receiver, it could not start transmission because it senses a busy medium, which introduces spatial reuse inefficiency. The binary exponential backoff scheme always favors the latest successful transmitter and results in unfairness.

The exposed station problem results from a situation where a transmission has to be delayed because of the transmission between two other stations within the sender's transmission range. In Fig. 2 we show a typical scenario where the exposed terminal problem occurs. Let us assume that A and C are within B's transmission range, and A is outside C's transmission range. Let us also assume that B is transmitting to A, and C has a frame to be transmitted to D. According to the carrier sense mechanism, C senses a busy channel because of B's transmission. Therefore, station C will refrain from transmitting to D, although this transmission would not cause interference at A. The exposed station problem may thus result in a reduction of channel utilization.

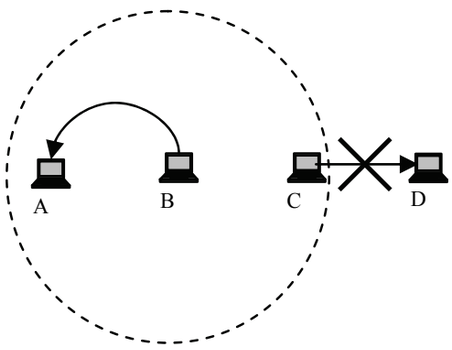


Fig. 2. Exposed terminal problem

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.

When TCP runs over 802.11 MAC, as pointed out, the instability problem becomes very serious. It is shown that collisions and the exposed terminal problem are two major reasons for preventing one node from reaching the other when the two nodes are in each other's transmission range. If a node cannot reach its adjacent node for several times, it will trigger a route failure, which in turn will cause the source node to start route discovery. Before a new route is found, no data packet can be sent out. During this process, TCP sender has to wait and will invoke congestion control algorithms if it observes a timeout. Serious oscillation in TCP throughput will thus be observed. Since large data packet sizes and back-to-back packet transmissions both decrease the chance of the intermediate node to obtain the channel, the node has to back off a random period of time and try again. After several failed tries, a route failure is reported.

2.3 Network partition

An ad hoc network can be represented by a simple graph *G*. Mobile stations are the "vertices." A successful transmission between two stations is an undirected "edge."

Network partition happens when *G* is disconnected. The main reason for this disconnection in MANETs is node mobility.
Mobility may induce link breakage and route failure between two neighboring nodes, as one mobile node moves out of the other's transmission range. Link breakage in turn causes packet losses. As we said earlier, TCP cannot distinguish between packet losses due to route failures and packet losses due to congestion. Therefore, TCP congestion control mechanisms react adversely to such losses caused by route breakages. Meanwhile, discovering a new route may take significantly longer time than TCP sender's RTO. If route discovery time is longer than RTO, TCP sender will invoke congestion control after timeout. The already reduced throughput due to losses will further shrink. It could be even worse when the sender and the receiver of a TCP connection fall into different network partitions. In such a case, multiple consecutive RTO timeouts lead to inactivity lasting for one or two minutes even if the sender and receiver finally get reconnected[9].
Another factor that can lead to network partition is energy constrained operation of nodes. An example of network partition is illustrated in Fig. 3. In this figure dashed lines are the links between nodes. When node D moves away from node C this movement, cause to network partition into two separate components. Clearly, the TCP agent of node A cannot receive the TCP ACK transmitted by node F. Originally, TCP does not have an indication about the exact time of network reconnection.

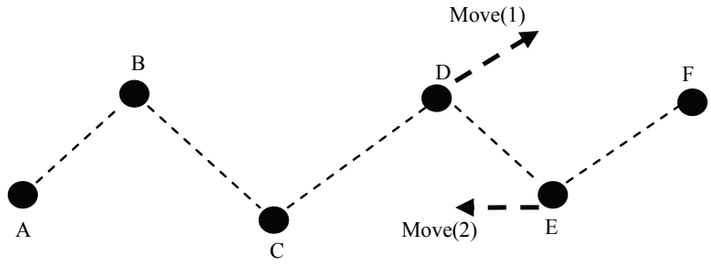


Fig. 3. Example for Network partition

This lack of indication may lead to long idle periods during which the network is connected again, but TCP is still in the backoff state.

2.4 Path asymmetry

Path asymmetry in ad hoc networks may appear in several forms as bandwidth asymmetry, loss rate asymmetry, and route asymmetry.
Bandwidth Asymmetry: Satellite networks suffer from high bandwidth asymmetry, resulting from various engineering tradeoffs (such as power, mass, and volume), as well as the fact that for space scientific missions, most of the data originates at the satellite and flows to the earth. The return link is not used, in general, for data transferring. For example, in broadcast satellite networks the ratio of the bandwidth of the satellite-earth link over the bandwidth of the earth-satellite link is about 1000 [10]. On the other hand, in ad hoc networks, the degree of bandwidth asymmetry is not very high. For example, the bandwidth ratio lies between 2 and 54 in ad hoc networks that implement the IEEE 802.11 version g protocol [11]. The asymmetry results from the use of different transmission rates. Because of this different transmission rates, even symmetric source destination paths may suffer from bandwidth asymmetry.

Loss Rate Asymmetry: This type of asymmetry takes place when the backward path is significantly more lossy than the forward path. In ad hoc networks this asymmetry occurs because packet losses depend on local constraints that can vary from place to place. Note that loss rate asymmetry may produce bandwidth asymmetry. For example, in multi-rate IEEE 802.11 protocol versions, senders may use the Auto- Rate-Fallback (ARF) algorithm for transmission rate selection [12]. With ARF, senders attempt to use higher transmission rates after consecutive transmission successes, and revert to lower rates after failures. So, as the loss rate increases the sender will keep using lower transmission rates.
Route Asymmetry: Unlike the previous two forms of asymmetry, where the forward path and the backward path can be the same, route asymmetry implies that distinct paths are used for TCP data and TCP ACKs. This asymmetry may be an artifact of the routing protocol used. Route asymmetry increases routing overheads and packet losses in the case of a high degree of mobility,1 because when nodes move, using a distinct forward and reverse route increases the probability of route failures experienced by TCP connections. However, this is not the case with static networks or networks that have a low degree of mobility, as in the case of a network with routes of high lifetime compared to the session transfer time. So it is up to the routing protocols to select symmetric paths when such routes are available in the case of ad hoc networks of high mobility.
In the context of satellite networks, there has been much research on how to improve TCP performance. However, since satellite networks are out of the scope of this article, we will limit ourselves to list three techniques introduced by these proposals, which we believe might be useful in ad hoc networks.

2.5 Routing failures

In wired networks route failures occur very rarely. In MANETs they are frequent events. The main cause of route failures is node mobility. Another factor that can lead to route failures is the link failures caused by the contention on the wireless channel, which is the main cause of TCP performance degradation in SANETs. The route reestablishment duration after route failure in ad hoc networks depends on the underlying routing protocol, mobility pattern of mobile nodes, and traffic characteristics. As already discussed, if the TCP sender does not have indications on the route re-establishment event, the throughput and session delay will degrade because of the large idle time. Also, if the new route established is longer or shorter in term of hops, than the old route TCP will face a brutal fluctuation in round trip time (RTT)[13].
In addition, in ad hoc networks, routing protocols that rely on broadcast Hello messages to detect neighbors' reachability may suffer from the "communication gray zones" problem. In these zones data messages cannot be exchanged, although broadcast Hello messages and control frames indicate that neighbors are reachable. So on sending a data message, routing protocols will experience routing failures.

2.6 Energy efficiency

As power is limited at mobile nodes, any successful scheme must be designed to be energy efficient. In some scenarios where battery recharge is not allowed, energy efficiency is critical for prolonging network lifetime[14]. Because batteries carried by each mobile node have limited power supply, processing power is limited. This is a major issue in ad hoc networks, as each node is acting as an end system and as a router at the same time, with the

implication that additional energy is required to forward and relay packets. TCP must use this scarce power resource in an "efficient" manner. Here, efficiency means minimizing the number of unnecessary retransmissions at the transport layer as well as at the link layer.2 In general, in ad hoc networks there are two correlated power problems: the first problem is "power saving," which aims at reducing power consumption; the second problem is "power control," which aims at adjusting the transmission power of mobile nodes[31]. Power saving strategies has been investigated at several levels of a mobile device, including the physical-layer transmissions, the operation systems, and the applications. Power control can be jointly used with routing or transport agents to improve the performance of ad hoc networks. Power constraints on communications also reveal the problem of cooperation between nodes, as nodes may not participate in routing and forwarding procedures in order to save battery power[32].

3. Current approaches to improving TCP performance in MANETs

In this section we present some schemes that have been proposed to improve TCP performance in ad hoc networks. There are some approaches for classifying these proposals that we introduce two of the most common of these classifying approaches. In first classifying scheme we classify these proposals in two categories: cross layer proposals and layered proposals. In layered proposals, the adaptation involves only one OSI layer, whereas in cross layer proposals at least two OSI layers are involved.
We classify layered proposals according to which layer the adaptation is done: at the TCP layer or at the link layer. On the other hand Cross layer proposals can be classified in three types :(1)TCP and network cross layer, (2)TCP and physical cross layer, and(3) network and physical cross layer.
Another classifying method can be as follow:

1. Modified TCP: This represents a class of transport layer approaches, where minor modifications are made to the TCP protocol to adapt it to the characteristics of an ad-hoc network, but the fundamental elements of TCP are still retained.
2. TCP aware Cross Layer Solutions: This represents a class of lower layer approaches that hide from TCP the unique characteristics of ad-hoc networks, and thus necessitate minimal changes to TCP. Such approaches can be used in tandem with the approaches in the previous class.
3. Ad-hoc Transport Protocols: Finally, this represents a class of new built-from-scratch transport protocols that are built specifically for the characteristics of an ad-hoc network, and are not necessarily TCP-like.

In the rest of this section we discuss in detail specific protocol instances of the different approaches and highlight the main features of each one. We classify these selected approaches in terms of usage from network layer feedback or not (using feedback means the proposal is cross layer solution). We terminate this section with describing a proposal that are not a modification from TCP but a new transport protocol that is suitable for ad hoc environments.

3.1 TCP with feedback solutions

Route changes are triggered by link breakages at some intermediate nodes (possibly the sender itself). Detecting these link Breakages is a basic requirement for any ad-hoc routing protocol. If the intermediate nodes, where the breakages happen, can convey this information back to the sender, the TCP controller at the sender will be able to detect the

event. We call this a network layer feedback mechanism. The majority of the existing approaches employ this detection mechanism, namely TCP-F (TCP-Feedback)[5], ELFN(Explicit Link Failure Notification)[16], ATCP (AdhocTCP)[7],and TCP-BuS[8].

3.1.1 TCP-F

TCP-F [15] relies on the network layer at an intermediate node to detect the route failure due to the mobility of its downstream neighbor along the route. A sender can be in an active state or a snooze state. In the active state, transport layer is controlled by the normal TCP. As soon as an intermediate node detects a broken route, it explicitly sends a route failure notification (RFN) packet to the sender and records this event. Upon reception of the RFN, the sender goes in to the snooze state, in which the sender completely stops sending further packets, and freezes all of its timers and the values of state variables such as RTO and congestion window size. Meanwhile, all upstream intermediate nodes that receive the RFN invalidate the particular route to avoid further packet losses. 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 node. Then it resumes the transmission from the frozen state.

3.1.2 TCP-ELFN

Holland and Vaidya proposed this feedback-based technique, the Explicit Link Failure Notification (ELFN)[16,19].The goal is to inform the TCP sender of link and route failures so that it can avoid responding to the failures as if congestion occurs. ELFN is based on the dynamic source routing (DSR)[20]routing protocol. To implement ELFN message, the route failure message of DSR is modified to carry a payload similar to the "host unreachable" ICMP (Internet Control Message Protocol) message. Upon receiving an ELFN, the TCP sender disables its congestion control mechanisms and enters in to a "stand-by" mode, which is similar to the snooze state of TCP-F mentioned above. Unlike TCP-F using an explicit notice to signal that a new route has been found, the sender, while on stand-by, periodically sends a small packet to probe the network to see if a route has been established. If there is a new route, the sender leaves the stand-by mode, restores its RTO and continues as normal. Recognizing most of popular routing protocols in ad hoc networks are on demand and route discovery/rediscovery is event driven, periodically sending a small packet at the sender is appropriate to restore routes with mild overhead and without modification to the routing layer.

3.1.3 ATCP protocol

ATCP [17] 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 maintain inter operation with non-ATCP machines. In particular, this approach relies on the ICMP protocol and 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, ATCP puts TCP in "persist 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, ATCP also puts the TCP in "persist mode" which only ends when the

connection is reestablished. At last, when network congestion is detected by the receipt of an ECN message, the ATCP 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 re ordering. In all cases the transfer time of a given file by ATCP 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 a simple ethernet networks connected in series in which each node had two ethernet cards. Moreover, 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.

3.1.4 TCP-BuS

TCP-BuS[18]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, these mechanisms all rely on the intermediate nodes, where the route Failures are detected, to send some control messages to notify the TCP sender. We categorize and call them the network layer feedback mechanisms.

3.2 TCP without feedback solutions

3.2.1 TCP-DOOR

TCP-DOOR [21] attempts to improve TCP performance by detecting and responding to out-of-order (OOO) packet delivery events and thus avoiding invoking unnecessary congestion

control by definition, OOO occurs when a packet sent earlier arrives later than a subsequent packet. In ad hoc networks, OOO may happen multiple times in one TCP session because of route changes. In order to detect OOO, ordering information is added to TCP ACKs and TCP data packets.OOO detection is carried out at both ends: the sender detects the Out-of-Order ACK packets and the receiver detects the Out-of-Order data packets. If the receiver detects OOO, it should notify the sender, considering the fact that it is the sender who takes congestion control actions. Once the TCP sender knows of an OOO condition, it may take one of the two responsive actions: temporarily disabling congestion control and instant recovery during congestion avoidance. The first action
means that, whenever an OOO condition is detected, TCP sender will keep its state variables such as RTO and the congestion window size constant for a time period T. The second action means that, if during the past time period T the TCP sender has already entered the state of congestion avoidance, and it should recover immediately to the state prior to such congestion avoidance. The main reason is the detection of OOO condition implies that a route change event has just occurred. However, OOO can be detected only after a route has recovered from failures. As a result, TCP-DOOR is less accurate and responsive than a feedback-based approach that is able to determine whether congestion or route errors occur, and hence report to the sender at the very beginning. Furthermore, it may not work well with multi-path routing since multi-path routing may cause OOO as well. Therefore, it is concluded that TCP-DOOR may work as an alternative to the feedback-based approach to improve TCP performance over ad hoc network, if the latter is not available.

3.2.2 Fixed RTO

Fixed RTO [22] is a very simple responding mechanism, originally coming from the consecutive time outs heuristic. If the sender encounters two consecutive Retransmission timeouts, it assumes some events other than congestion happen. Then the Value of retransmission timeout is fixed, without incurring exponential backoff. The RTO Remains fixed until the route is re-established and the retransmitted packet is acknowledged. This simple technique is particularly effective when network partition happens. Without fixing the RTO, it will become longer and longer exponentially, which implies that the chance to probe a valid route is smaller and smaller. An improved approach is, not only to fix the RTO, but also to reset it to the initial value which is a short time period. In other words, it is better to probe the network frequently after a network partition is believed to have happened in order to avoid wasting time idling.

3.3 Ad-hoc transport protocols

In this section we describe a novel transport protocol for MANETs. Unlike other proposals, this protocol is not a modification of the TCP but is specifically tailored to the characteristics of the MANET environment. It is able to manage efficiently route changes and route failures. Furthermore, it includes a completely re-designed congestion control mechanism. Finally, it is designed in such a way to reduce as much as possible the number of useless retransmissions. This is extremely important since retransmissions consume energy.

3.3.1 ATP (Ad hoc Transport Protocol)

ATP (ad-hoc transport protocol) is tailored toward the characteristics of ad-hoc networks. ATP, by design, is an antithesis of TCP and consists of: rate based transmissions, quick-start

during connection initiation and route switching, network supported congestion detection and control, no retransmission time outs, decoupled congestion Control and reliability, and coarse grained receiver feedback. Briefly, just as in TCP, ATP primarily consists of mechanisms at the sender to achieve effective congestion control and reliability. However, unlike in TCP, ATP relies on feedback not just from the receiver, but also from the intermediate nodes in the connection path. In terms of specific functionality, the intermediate nodes provide congestion feedback to the sender, while the receiver provides feedback for both flow control and reliability. The receiver also acts as a collator of the congestion information provided by the intermediate nodes in the network before the information is sent back to the sender. The receiver provides the reliability, flow control, and collated congestion control information through periodic messages. The sender on the other hand, is responsible for connection management, start-up rate estimation (with network feedback), congestion control, and reliability.

4. ADHOCTCP

In this section for description o f new proposed approach we first determine the network states that TCP must monitor. Identifying three network states is necessary to improve TCP performance over ad hoc networks that states are: CONGESTION, CHANNEL ERROR, and DISCONNECTION. These states should be our identification target. We use end-to-end measurements to identify the presence of congestion in the network; we must then determine what available end-to-end metrics can be used to accurately identify congestion state in the network. The goal of the identification algorithm is therefore a mapping from metric measurements to the target states that in ADHOCTCP we describe the identification algorithm to decide that network is congested or not. We first assume a situation in which TCP knows why its packets are being lost and consider what TCP should do to improve its performance. First, if the packet loss is due to congestion, TCP should apply the congestion control mechanisms; but if not, TCP might do better not to slow down and exponentially backoff its retransmission timeout. Therefore knowing whether the current state of network is congested or not is important. As it turns out, proper congestion identification proves to be the biggest improvement to TCP in ad hoc networks. Second, if the packet is lost due to reasons other than congestion, TCP can benefit if it further knows whether the loss is due to channel errors or network disconnection. If the loss is due to channel errors, a simple retransmission is adequate. However, if it is due to disconnection, some special probing mechanisms might be needed for a prompt transmission recovery upon network reconnection.

4.1 Congestion

TCP attempt to fully utilize the network bandwidth makes ad hoc networks easily go into congestion. In addition, due to many factors such as route change and unpredictable variable MAC delay, the relationship between congestion window size and the tolerable data rate for a route is no longer maintained in ad hoc networks. The congestion window size computed for the old route may be too large for the newly found route, resulting in network congestion if the sender still transmits at the full rate allowed by the old congestion window size congestion/overload may give rise to buffer overflow and increased link contention, which degrades TCP performance. As a matter of fact, [23] showed the capacity of wireless ad hoc networks decreases as traffic and/or competing nodes arise.

When network congestion occurs, ad hoc transport should adopt the same congestion control actions as conventional TCP [24]. Here, we define congestion as queue build-up and packets being dropped due to buffer overflow at some nodes.

4.1.1 Identifying congestion

There are two types of approaches in detecting network congestion in the Internet. One is based on end-to-end measurement and the other on feedback from intermediate gateways in the network. Standard TCP [25] uses end-to-end measurement of RTT and packet loss to detect congestion; RED/ECN [26] provides congestion notification by monitoring the instantaneous queue size at the network gateways.

The end-to-end approach is easy to implement and deploy, requires no network support, and provides the flexibility for backward compatibility. However, using single metric measurements, the probability of false congestion detection in an uncongested ad hoc network is quite high. This sort of false detection can lead to serious throughput degradation. In ADHOCTCP we use of multi-metric joint identification for identifying congestion in ad hoc networks. By exploiting the degree of independence in measurement noise of individual metrics, the probability of false identification can be significantly reduced by cross-verification.

Two metrics are devised to detect congestion, IDD (Inter Delay Difference) and STT (Short Term Throughput). They each exhibit a unique pattern upon congestion; and in non-congestion states, they are influenced by different network conditions in such a way that their respective measurement noise is largely independent.

Inter-packet delay difference (*IDD)* Metric: IDD measures the delay difference between consecutive packets that calculate as fallow:

$A^{i+1} - A^i - (S^{i+1} - S^i)$, where A^i is the arrival time of packet i and S^i is its sending time from the sender

It reflects the congestion level along the forwarding delivery path by directly sampling the transient queue size variations among the intermediate nodes.

Short-term throughput (*STT):* STT metric calculate as fallow:

Np(T)/T , where Np(T) is the # of received packets during interval T

Compared with IDD, STT is also intended for network congestion identification. It provides observation over a time interval T, and is less sensitive to short term out-of-order packet delivery than IDD. Therefore, STT is more robust to transient route changes, which can be very frequent in a mobile ad hoc network. However, using STT alone to detect network congestion can be susceptible to measurement noise introduced by bursty channel error, network disconnections or altering TCP source rates. We combine STT and IDD to jointly identify network congestion.

We identify a congestion state when both IDD is HIGH and STT is LOW, and non-congestion state if otherwise. We define a value to be HIGH or LOW if respectively it is within the top or bottom 30% of all samples.

The identification module is plugged into the receiver side. Space is allocated for storing metrics samples. Identifying high or low related calculations are performed after normal processing of each incoming data packet. One bit used for representing congestion state. We introduce an option field in the TCP header and set the corresponding bit in each outgoing ACK packet. Algorithm 1 shows receiver side algorithm:

Algorithm1: Receiver Side Algorithm: Upon packet arrival
1: process data and generate ACK packet 2: compute sample value for two metrics 3: estimate HIGH/LOW for each metric 4: network state identification (congested or not) 5: set state bit in option field of out-going ACK packet 6: transmit ACK

Logic to process the ADHOCTCP option field of a TCP header is introduced at the sender side to read in this bit from incoming ACK packets. We extend the code that handles third duplicate ACKs and retransmission timeouts to follow the ADHOCTCP design. In particular, when a sender goes into the probing state, it caches its current transmission state and begins using small packets (8 bytes payload) to probe the receiver until it receives an acknowledgement of the reception of the probing packet. Upon leaving the probing state, the previous transmission state is then restored.

4.2 Disconection

4.2.1 Impact of mobility

Mobility may induce link breakage and route failure between two neighboring nodes, as one mobile node moves out of the other's transmission range. Link breakage in turn causes packet losses and TCP cannot distinguish between packet losses due to route failures and packet losses due to congestion. Therefore, TCP congestion control mechanisms react adversely to such losses caused by route breakages [18,22,27]. Meanwhile, discovering a new route may take significantly longer time than TCP sender's RTO. If route discovery time is longer than RTO, TCP sender will invoke congestion control after timeout. The already reduced throughput due to losses will further shrink. It could be even worse when the sender and the receiver of a TCP connection fall into different network partitions. In such a case, multiple consecutive RTO timeouts lead to inactivity lasting for one or two minutes even if the sender and receiver finally get reconnected. Fu et al. conducted simulations considering mobility, channel error, and shared media-channel contention [4]. They indicated that mobility-induced network disconnections and reconnections have the most significant impact on TCP performance comparing to channel error and shared media-channel contention. As mobility increases, compared to a reference TCP, TCP NewReno suffers from a relative throughput drop ranging from almost 0% in a static case to 90% in a highly mobile case (when moving speed is 20m/s). In contrast, congestion and mild channel error (say 1%) have less visible effect on TCP (with less than 10% performance drop compared with the reference TCP).

4.2.2 Disconnection identification and reaction

It is likely that the ad hoc network may periodically get partitioned for several seconds at a time. If the sender and the receiver of a TCP connection lie in different partitions., all the sender's packets get dropped by the network resulting in the sender invoking congestion control. If the partition lasts for a significant amount of time (say several times longer than the RTO), the situation gets even worse because of a phenomena called *serial timeouts*.

The goal is to inform the TCP sender of link and route failures so that it can avoid responding to the failures as if congestion occurs. Our disconnection identification method is similar to TCP-ELFN [16], we use from EPLN (Explicit Packet Loss Notification) for inform sender. EPLN is based on the dynamic source routing (DSR) [10] routing protocol. To implement EPLN message, the route failure message of DSR is modified to carry a payload similar to the "host unreachable" ICMP (Internet Control Message Protocol) message. Upon receiving an EPLN, the TCP sender disables its congestion control mechanisms and enters into a "stand-by" mode, the sender, while on stand-by, periodically sends a small packet to probe the network to see if a route has been established. If there is a new route, the sender leaves the stand-by mode, restores its RTO and continues as normal.

4.3 Channel error

Bursty bit errors may corrupt packets in transmission, leading to the loss of TCP data packets or acknowledgments (ACKs). If it cannot receive the ACK within the retransmission timeout (RTO), the TCP sender immediately reduces its congestion window to one packet, exponentially backs off its retransmission, and retransmits the lost packet. Intermittent channel errors may thus cause the congestion window size at the sender to remain small, resulting in low throughput.

If RTO expires or sender receives 3 duplicate ack and network state does not detect as congestion by receiver's end to end measurements, sender assume that packet loss is due to channel error. In such case because packet loss is a random packet loss, without slowing down, the sender will re-transmit the lost packet [3][28].

5. Performance evaluation

We used *ns-2* [29] network simulator with Monarch Project's wireless and mobile extensions [30]. The network interface model provides a 2Mbps transmission rate and a nominal transmission range of 250m; the network interface uses IEEE 802.11 DCF MAC protocol [26]. The mobility model is *random waypoint model* in a rectangular field. In this model, a node starts at a random position, picks a random destination, moves to it at a randomly chosen speed, and pauses for a specified pause time. The node speed was randomly chosen from v m/s, where v is node mean speed. We used pause time 0 s for all simulations. The two field configurations we used were 1500m*1000m field with 50 nodes and 2200m*600m field with 100 nodes. We used TCP-NewReno with the packet size of 1460 bytes. The maximum size of both congestion window and receiver's advertised window is 8. FTP is the application that we used over TCP.

5.1 Simulation results

TCP's congestion window never really has been opportunity to grow in size because losses due to bit error result in congestion control. ADHOCTCP's congestion window on the other hand, never shrinks. This accounts for the dramatic difference in TCP and ADHOCTCP performance in Figure 6. TCP's congestion windows remains small making TCP behave almost like a stop-and-wait protocol (figure 4).

The first experiments we ran did not include disconnection or congestion events. The connection was only subjected to bit error that occurred at a BER of 10^{-5} at each hop.

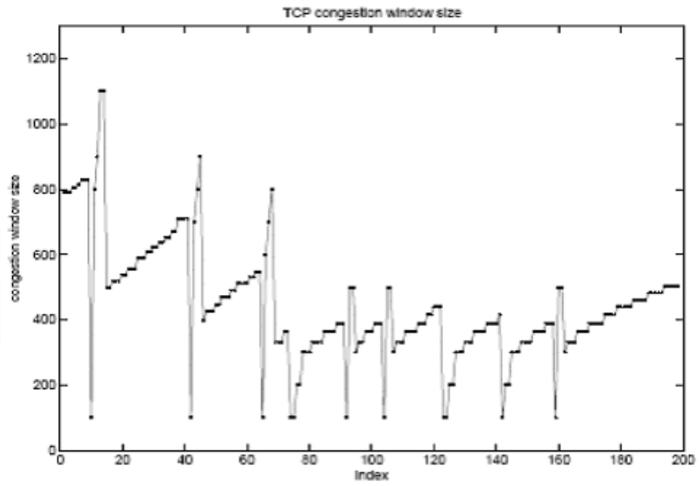


Fig. 4. TCP congestion window in presence of Bit error only

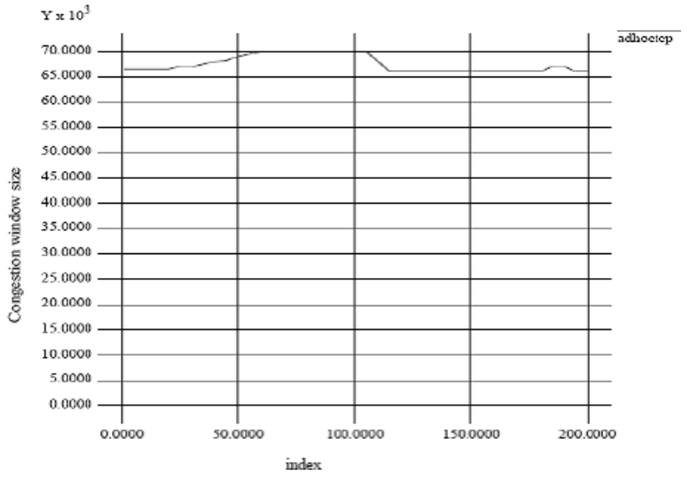


Fig. 5. ADHOCTCP congestion window in presence of bit error only

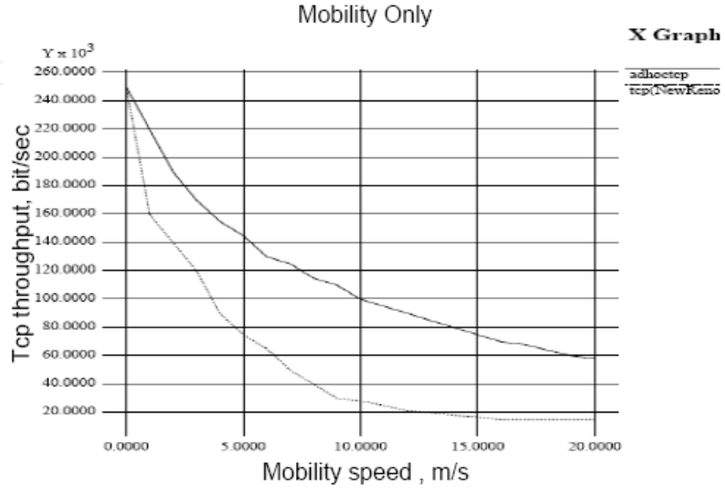


Fig. 6. ADHOCTCP performance in the presence of node mobility

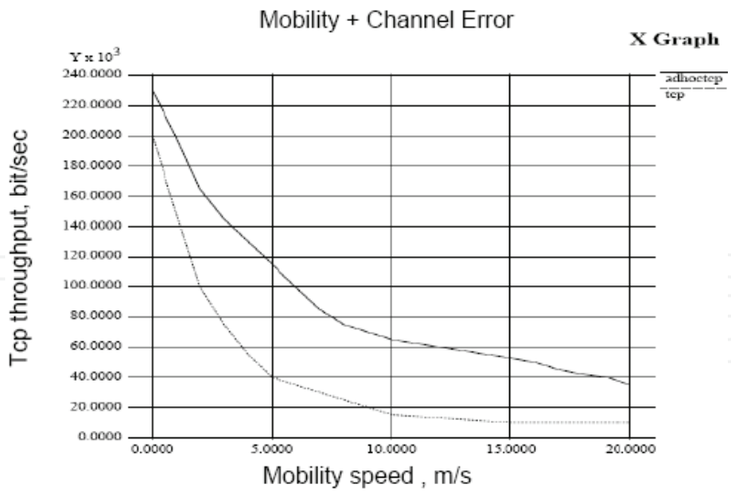


Fig. 7. ADHOCTCP performance in presence of node mobility and channel error

In the next experiment, we introduced periodic congestion in the network that results presented in figure8.

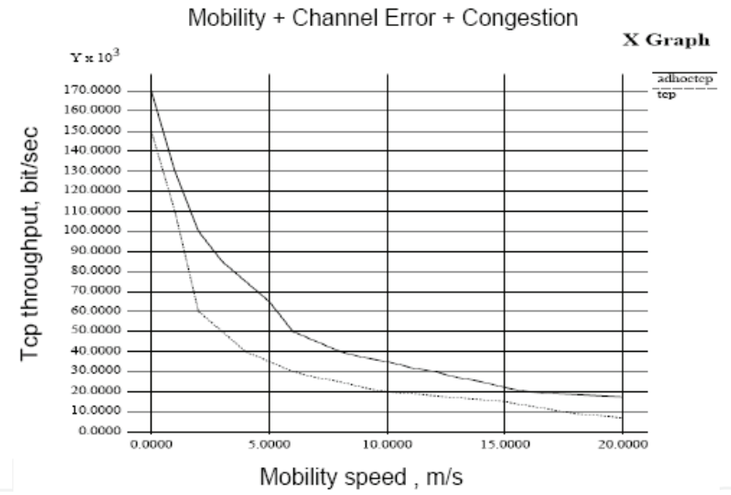


Fig. 8. ADHOCTCP performance in presence of mobility, channel error and congestion

There are a couple of reasons for the difference in performance between TCP and ADHOCTCP .First, the number of time out events in TCP is high because of the high bit error as well as because of loss due to congestion. Thus, TCP does not get much of an opportunity to grow its congestion window. ADHOCTCP, on the other hand, defers to TCP's congestion only when network state identified as congestion by receiver side. In other cases, it retransmits the lost packets from TCP's buffer.

6. Conclusions and future research

As the assumption made by TCP that any packet loss is due to network congestion is not valid in ad hoc networks, either TCP should be capable of distinguishing various reasons of packet losses or such non-congestion related losses should be reduced. To enable TCP to

identify various causes of packet losses, there are largely two approaches, depending on whether or not network feedback information is used. Feedback-based schemes seem to be able to react more quickly to non-congestion related packet losses, thus to be more effective in enhancing TCP performance. However, the price to be paid is that they are more difficult to implement, since they require end nodes and intermediate nodes to cooperate with each other. On the other hand, approaches without feedback are relatively simple to implement. However, the performance gain may not be high enough.

This paper explores an alternative approach that relies solely on end-to-end mechanisms. To robustly detect congestion state in the presence of measurement noise, we propose a multiple-metric based joint detection technique. In this technique, a congestion event is signaled only if all the relevant metrics detect it. Our simulations show that ADHOCTCP is able to significantly reduce the probability of false detection while keeping the incompatible detection errors low and thus greatly improves the transportation performance in a TCP friendly way. This demonstrates that the end-to-end approach is also viable for ad hoc networks.

7. References

[1] W. Richard Stevens TCP/IP *Illustrated, Volume 1, The Protocols*, AWL, 1994

[2] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *MOBICOM'99*.

[3] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, "Improving TCP/IP performance over wireless networks," *MOBICOM'95*.

[4] S. Biaz and N.H. Vaidya, "Distinguishing congestion losses from wireless transmission losses" *IEEE 7th Int. Conf. on Computer Communicationsand Networks*, October 1998.

[5] IEEE 802.11WG, Part 11:Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, Standard, Aug. 1999.

[6] F. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II — The hidden Terminal Problem in Carrier Sense Multiple-Access Modes and the Busy-Tone Solution," IEEE Trans. Net., vol. 23, no. 12, 1975, pp. 1417–33.

[7] Z. Fu et al., "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," Proc. IEEE INFOCOM, San Francisco, USA, Apr. 2003.

[8] K. Xu, M. Gerla, and S. Bae, "Effectiveness of RTS/CTS Handshake in IEEE 802.11-Based Ad Hoc Networks," Ad Hoc Net. J., Elsevier, vol. 1, no. 1, July 2003, pp. 107–23.

[9] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," RFC 2988, Category: Standard Track, Nov. 2000.

[10] R. Durst, G. Miller, and E. Travis, "TCP Extensions for Space Communications," Proc. ACM MOBICOM, Rye, NY, 1996, pp. 15-26.

[11] "IEEE 802.11 WLAN standard," Web site: http://standards. ieee.org/getieee802

[12] A. Kamerman and L. Monteban., "Wavelan II: A High-Performance Wireless LAN for the Unlicensed Band," Bell Labs Tech. J., Summer 1997, pp. 118–33.

[13] H. Lundgren, E. Nordstro, and C. Tschudin, "Coping with Communication Gray Zones in IEEE 802.11b-Based Ad Hoc Networks," Proc. ACM Wksp. Wireless Mobile Multimedia, Atlanta, GA, USA, Sept. 2002, pp. 49–55.

[14] C. Jones et al., "A Survey of Energy Efficient Network Protocols for Wireless and Mobile Networks," ACM Wireless Net., vol. 7, no. 4, 2001, pp. 343–58.

[15] K.Chandran, S.Raghunathan, S.Venkatesan, R.Prakash. A Feedback Based Scheme For Improving TCP Performance In AdHoc Wireless Networks. In Proceedings of International Conference on Distributed Computing Systems- ICDCS '98. pp. 472-479,1997.

[16] G.HollandandN.H.Vaidya,"Analysis of TCP performance over mobile ad hoc networks,"ACMMOBICOM'99, Seattle, August 1999.

[17] J. Liu and S. Singh. ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1300{1315, July 2001.

[18] D.Kim,C.-K.Toh,andY.Choi. TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks. Journal of Communications and Networks, Vol. 3, No. 2. Jun. 2001.

[19] J.P.Monks, P.Sinhaand V.Bharghavan, "Limitations of TCP-ELFN for ad hoc networks,"MOMUC2000

[20] D. Johnson, D. Maltz, Y.-C. Hu, and J. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks (DSR). IETF Internet-Draft, draft-ietf-manet-dsr-06.txt, work in progress, Nov.2001

[21] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response," MobiHoc'02, pp. 217-225, Lausanne, Switzerland, Jun 2002.

[22] T.Dyer and R.Boppana. A comparison of TCP performance over three routing protocols for mobile Ad hoc networks. In Proceedingsofthe2001ACM International Symposium on Mobile Ad Hoc Networking & Computing(MobiHoc'01), Long Beach, California, Oct. 2001.

[23] J. Li, C. Blake, D. S. J. De Couto, H. Lee, and R. Morris, "Capacity of ad hoc wireless networks," *ACM MobiCom'01*, Rome, Italy, July 2001.

[24] M.Allman, V.Paxson, and W. Stevens, "TCP Congestion Control" *RFC 2581* April 1999.

[25] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5):8-23, Oct. 1994.

[26] S. Floyd, "TCP and explicit congestion notification," *ACM CCR*, 1994

[27] A. Ahuja, S. Agarwal, J. P. Singh and R. Shorey, "Performance of TCP over different routing protocols in mobile ad-hoc networks," *IEEE VTC 2000*, vol. 3, pp. 2315-2319, Tokyo.

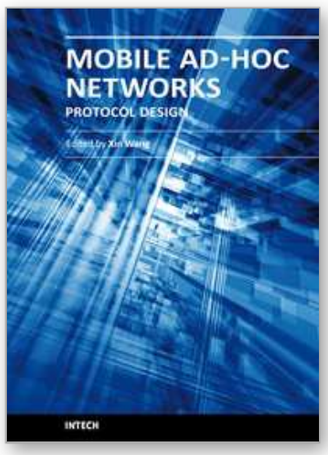
[28] H. Balakrishnan, and R. Katz, "Explicit loss notification and wireless web performances," *Globecom'98*.

[29] K. Fall and K. Varadhan, *ns* notes and documentation, LBNL (August 1998) http://www-mash.cs.berkeley.edu/ns/

[30] J. Broch, D.A. Maltz, D.B. Johnson, Y. Hu and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: *ACM/IEEE International Conference on Mobile Computing and Networking* October 1998) pp. 85–97.

[31] H. Singh and S. Singh, "Energy consumption of TCP reno, newreno, and SACK in multi-hop wireless networks," ACM SIGMETRICS'02, Jul. 2002.

[32] I. Ali, R. Gupta, S. Bansal, A. Misra, A. Razdan and R. Shorey, "Energy efficiency and throughput for TCP traffic in multi-hop wireless networks," IEEE INFOCOM' 02, New York, 2002.



Mobile Ad-Hoc Networks: Protocol Design

Edited by Prof. Xin Wang

ISBN 978-953-307-402-3
Hard cover, 656 pages
Publisher InTech
Published online 30, January, 2011
Published in print edition January, 2011

Being infrastructure-less and without central administration control, wireless ad-hoc networking is playing a more and more important role in extending the coverage of traditional wireless infrastructure (cellular networks, wireless LAN, etc). This book includes state-of-the-art techniques and solutions for wireless ad-hoc networks. It focuses on the following topics in ad-hoc networks: quality-of-service and video communication, routing protocol and cross-layer design. A few interesting problems about security and delay-tolerant networks are also discussed. This book is targeted to provide network engineers and researchers with design guidelines for large scale wireless ad hoc networks.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Seyed Mohsen Mirhosseini and Fatemeh Torgheh (2011). ADHOCTCP: Improving TCP Performance in Ad Hoc Networks, Mobile Ad-Hoc Networks: Protocol Design, Prof. Xin Wang (Ed.), ISBN: 978-953-307-402-3, InTech, Available from: http://www.intechopen.com/books/mobile-ad-hoc-networks-protocol-design/adhoctcp-improving-tcp-performance-in-ad-hoc-networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen