4TH QUARTER 2006, VOLUME 8, NO. 4

IEEE COMMUNICATIONS SURVEYS
The Electronic Magazine of Original Peer-Reviewed Survey Articles

www.comsoc.org/pubs/surveys

TRANSMISSION CONTROL PROTOCOL (TCP) IN WIRELESS NETWORKS: ISSUES, APPROACHES, AND CHALLENGES

KA-CHEONG LEUNG AND VICTOR O. K. LI, THE UNIVERSITY OF HONG KONG

ABSTRACT

The Transmission Control Protocol (TCP) is the most popular transport layer protocol for the Internet. Due to the characteristics specific to wireless networks, such as signal fading and mobility, packets may be lost due to congestive and noncongestive losses. Substantial noncongestive losses violate the design principles of some traffic control mechanisms in TCP and thus pose performance problems. In this article we provide a comprehensive and in-depth survey on recent research in TCP for wireless communications. The taxonomy and characteristics of wireless networks, and problems for TCP in wireless communications are introduced. Various representative algorithms which preserve the end-to-end semantics are examined. Some open questions are discussed in order to stimulate further research in this area.

The Internet provides a convenient and cost-effective communication platform for electronic commerce, education, and entertainment. The success of the Internet stems from its capabilities to support survivable, robust, and reliable end-to-end data transfer services for a myriad of applications running over a set of end-systems. The Internet originated from the Advanced Research Projects Agency Network (ARPANET) designed to support survivable military communications. Currently, the Transmission Control Protocol (TCP) [1] is the most popular transport layer protocol for point-to-point, connection-oriented, in-order, reliable data transfer in the Internet. TCP is the de facto standard for Internet-based commercial communication networks.

INTRODUCTION TO TCP

TCP is a byte-stream protocol; its flow control and Acknowledgment are based on byte number rather than packet number [2]. However, the smallest unit of data transmitted in the Internet is a data segment or packet, each identified by a data octet number. When a destination receives a data segment, it acknowledges the receipt of the segment by issuing an Acknowledgment (ACK) with the next expected data octet number. The time elapsed between when a data segment is sent and when an ACK for the segment is received is known as the round-trip time (RTT) of the communication between the source and the destination, which is the sum of the propagation, transmission, queuing, and processing delays at each hop of the communication, and the time taken to process a received segment and generate an ACK for the segment at the destination.

The flow control mechanism used by TCP is a credit allocation scheme. To avoid overwhelming its buffer space, a destination advertises to the associated source the size of a window (advertised window), which indicates the number of data bytes beyond the acknowledged data the source can send to the destination. This information is included in the header of each TCP (data or control) segment sent to the source. Suppose the source knows that, based on ACK(s) received, Byte x is the last data byte received by the destination. The source can send data up to Byte $x + W$, where W is the size of the advertised window. An example of the source sequence number space is exhibited in Fig. 1.

CONGESTION CONTROL OPERATIONS OF TCP

To achieve good performance, it is necessary to control network congestion so that the number of packets within the Internet is below the level at which the network performance drops significantly. Various congestion control measures [3] have been implemented in TCP to limit the sending rate of data entering the Internet by regulating the size of the congestion window *cwnd*, the number of unacknowledged segments allowed to be sent. These measures include slow start, conges-

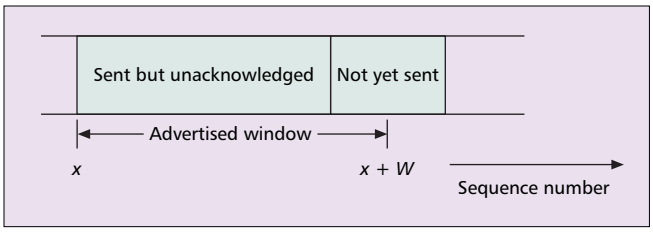

Figure 1. *An illustration of the source sequence number space and advertised window.*

tion avoidance, fast retransmit, and fast recovery. When a new connection is established, TCP sets *cwnd* to one. In slow start, the value of *cwnd* is incremented by one each time an ACK is received until it reaches the slow start threshold, *ssthresh*.

TCP uses segment loss as an indicator of network congestion. A retransmission timer is associated with each transmitted segment and a timer expiration before receiving an ACK signals a segment loss. The retransmission timeout period (RTO) is determined by the sum of the smoothed exponentially weighted moving average and a multiple of the mean deviation of RTT [4]. When a timeout occurs, *ssthresh* is set to half of the amount of outstanding data sent to the network. The slow start process is performed starting with *cwnd* being set to one. The congestion avoidance phase is then carried out where *cwnd* is increased by one for each RTT.

When the data octet number of an arriving segment is greater than the expected one, the destination finds a gap in the sequence number space (known as a sequence hole) and thus immediately sends out a duplicate ACK, that is, an ACK with the same next expected data octet number in the cumulative Acknowledgment field,[1] to the source. If the communication channel is an in-order channel, the reception of a duplicate ACK implies the loss of a segment. When the source receives three duplicate ACKs, fast retransmit is triggered such that the inferred loss segment is retransmitted immediately, before the expiration of the retransmission timer.

Fast recovery works as a companion of fast retransmit. A fast retransmission suggests the presence of mild network congestion. *ssthresh* is set to half of the amount of outstanding data sent to the network. Since the reception of a duplicate ACK indicates the departure of a segment from the network, *cwnd* is set to the sum of *ssthresh* and the number of duplicate ACKs received. When an ACK for a new segment arrives, *cwnd* is reset to *ssthresh* and congestion avoidance is triggered.

TCP Tahoe [5] and TCP Reno [3] are the two most popular TCP variants in the Internet. TCP Tahoe includes slow start, congestion avoidance, and fast retransmit,[2] whereas TCP Reno adds fast recovery to the congestion control mechanisms in TCP Tahoe so that fast recovery works in conjunction with fast retransmit.

These proposed congestion control mechanisms have implicitly assumed that all segment losses are congestive losses, that is, packets are dropped by routers due to network congestion. Any packet loss not due to network congestion is a noncongestive loss, and such segment losses are assumed to be negligible. Unfortunately, this is no longer valid for a connection with a wireless link. In wireline communications, signals are propagated over a point-to-point, wired medium, such as optical fiber and coaxial cables. Packets are seldom lost during transmission. Wireless transmissions are broadcast in nature, and they share and contend for the same transmission medium. Their underlying radio signals may interfere with each other. Besides, the radio signal strength can be distorted or weakened because of signal fading.[3] This unreliable nature of the wireless medium causes a substantial number of packet losses, exceeding the tolerable loss limit of TCP. This results in the violation of the design principles of some traffic control mechanisms in TCP and thus poses performance problems.

Overview of the Article

The objective of this article is to present a clear overview of recent developments and explore some open research issues and challenges on TCP in wireless networks. We survey the end-to-end solutions proposed to date that require no intermediaries to scoop the state of a connection for TCP in wireless networks. Some of these solutions may require supporting functions implemented at the routers for the sake of efficiency and performance enhancements. These solutions preserve the end-to-end semantics of any established TCP connection so that the traffic control and maintenance of the connection are performed by the two end systems of the connection, based on the measured network conditions. This follows the basic design tenet of the Internet and TCP guided by the end-to-end arguments [6]. Any modifications that break the end-to-end semantics of TCP, such as splitting a TCP connection into the wired and wireless portions so that the traffic control is done separately, cannot guarantee the arrival of a certain data segment at the destination and thus maintain the end-to-end data delivery, even though the source has received the ACK of that segment. Besides, a solution that needs some intermediaries or agents to scoop the state of a connection is intrusive, since some packets belonging to the connection are inspected or/and even cached by these intermediaries to infer the end-to-end connection state. The scheme does not even work, say, if a TCP segment is encrypted end-to-end by its source. Hence, any solutions that require scooping intermediaries or do not preserve the end-to-end semantics of a TCP connection are not discussed here. Interested readers can refer to [7] for early work to improve the performance of TCP in wired-cum-wireless environments. They can also refer to [8, 9] for surveys of recent TCP enhancements in ad hoc networks and last hop wireless networks, respectively.

Another contribution of the article is to give the readers a new angle from which to view the existing state of the art by classifying the surveyed solutions based on the way they tackle the problems in wireless networks. In [7], the authors categorized the proposed mechanisms as the link-layer solutions, split solutions, TCP modifications, new transport protocols, and wireless application protocol. In [8], the authors classified the surveyed proposals according to which protocol layer(s) the enhancements have actually been implemented. Our article focuses on enhancements that have been implemented in the TCP clients that can allow readers to better comprehend how the existing solutions alleviate the wireless problems through TCP itself. In [9], the authors grouped the solutions based on whether a connection is split between wired and wireless portions as well as the way a wireless segment loss is

[1] *A cumulative ACK is an ACK that uses the cumulative ACK field in the TCP header to acknowledge all in-sequence data received by the destination.*

[2] *After fast retransmit is triggered in TCP Tahoe,* ssthresh *is set to half of the amount of outstanding data sent to the network. Slow start is then carried out with* cwnd *being set to one.*

[3] *Signal fading refers to the attenuation or distortion of a signal due to propagation loss, and reflection, diffraction, and scattering caused by obstacles.*

handled. However, the categories are not mutually exclusive. There exist some proposed solution, such as ATCP [10], that can be put into more than one categories, say, the end-to-end connection and explicit notification protocols for ATCP. Nevertheless, the proposed categories in our article are mutually exclusive so that this can give readers a clear picture how the existing work can be categorized.

Furthermore, this survey article provides the readers a short tutorial of the surveyed representative solutions so that they can understand the basic mechanisms of these enhancements easily. In other words, our intended readers are the general audience who would like to quickly acquire the state of the art on TCP solutions in wireless networks.

The rest of the article is organized as follows. First, we identify the issues of running TCP on wireless networks. We give the taxonomy and characteristics of the wireless networks. We then discuss the problems due to the unique nature of the wireless medium, and provide a taxonomy of the existing solutions to the problems. We present a survey of the recently proposed solutions that preserve the end-to-end semantics and require no scooping intermediaries. Further discussion of the algorithms and some open research issues and challenges are given. Finally, we summarize and conclude the article.

Issues of Running TCP on Wireless Networks

In this section we first give the taxonomy of the wireless networks, namely, the infrastructured networks and the ad hoc networks. The characteristics of wireless networks are then identified. Afterwards, we discuss the problems of running TCP on a wireless medium.

Taxonomy of Wireless Networks

The distinguishing feature of wireless networks is that packets (or segments) are transmitted with the presence of wireless links. In wireline networks, two devices can communicate directly only when there is a wired link connecting them. In other words, a device can send messages in a wireless network via the wireless medium, air, to another device provided that the receiver is within the transmission range[4] of the sender. This adds flexibility to how a wireless network is formed and structured. Besides, it supports device mobility. There are two major types of wireless networks, namely, the infrastructured networks and the ad hoc networks. These will be described in detail next.

Infrastructured Networks — An infrastructured network is one with planned, permanent network device installations. It can be set up with a fixed topology, to which a wireless host can connect via a fixed point, known as a base station or an access point. The latter is connected to the backbone network, often via a wired link. Cellular networks [11] and most of the wireless local area networks (WLANs) [12] operate as the static infrastructured networks. All wireless hosts within the transmission coverage of the base station can connect to it and use it to communicate with the backbone network. This means that all communications initiated from or destined to a wireless host have to pass through the base station to which the host connects directly.

In addition, an infrastructured network is also be established with a quasi-static or a dynamic topology. A satellite network [13] belongs to this category. It has a space segment and a ground segment. The space segment comprises of satellites. The ground segment has a number of base stations, also known as gateway stations (GSs), through which all communications via long-haul satellite links take place.

The base station, or access point, is a critical element for communication. To maintain an ongoing connection when a mobile host moves away from the coverage of its base station, a terminal handoff occurs such that a mobile host hands over its proxy for communication from one base station to another one. Whenever the coverages of several neighboring base stations overlap with each other, a mobile host may connect to one of the reachable base stations based on certain criteria.

Ad Hoc Networks — An ad hoc network, such as a packet radio network, is one without a fixed topology. A wireless host can freely communicate with another host directly whenever the receiver is in its transmission coverage. If a wireless host would like to send messages to another host which is not in the coverage region, it will first relay them to a host in its transmission range. The host functions as a relay to forward the messages on its way to the destination.

The major advantage of this configuration is flexibility. An ad hoc network can be built easily, without the need of any preset, fixed infrastructure. In addition, an ad hoc network is generally more robust than an infrastructured network as it does not have any critical device to maintain the network connectivity. In other words, it is unlikely an ad hoc network will be partitioned due to the failure of a wireless host, but the malfunction of a base station may partition an infrastructured network, blocking the communication between all wireless hosts connecting to the failed base station and all other hosts in the network.

However, there are some drawbacks for ad hoc networks. First, it is much more difficult and complex to perform routing in ad hoc networks because of frequent changes in the network topology due to host mobility. Second, it is more difficult to control or coordinate proper operation of an ad hoc network, since each wireless host may have its own algorithms to perform activities such as time synchronization, power management, and packet scheduling. In an infrastructured network, these algorithms are often implemented in and thus harmonized by the base stations or access points.

Characteristics of Wireless Networks

There are four major characteristics of wireless networks: channel contention, signal fading, mobility, and limited power and energy.

Channel Contention — In a wireless network, signals are broadcast and may interfere with each other. A collision will be sensed and transmissions may fail when there exists concurrent transmissions within the interference range[5] of either sender. Thus, a medium access protocol is needed to coordinate the transmission accesses of the wireless channel so as to achieve a reasonably high channel utilization and goodput.

The channel contention problem is exacerbated in time division multiple access (TDMA)[6] based multihop wireless networks. The number of segments that can be in flight con-

[4] The transmission range or coverage of a device is defined as the region in which another device can successfully receive information sent by the preceding device.

[5] The interference range of a device is defined as the region in which a transmission initiated by the device will interpose or corrupt other ongoing transmissions.

currently is limited from a source to a destination, thereby constraining the achievable throughput for a TCP connection. Furthermore, the correlated arrivals of data segments and their ACKs lead to contention for the wireless channel, causing excessive collisions and packet losses [14].

Signal Fading — Unlike wired media, signals transmitted over a wireless medium may be distorted or weakened because they are propagated over an open, unprotected, and ever-changing medium with irregular boundary. Besides, the same signal may disperse and travel on different paths due to reflection, diffraction, and scattering caused by obstacles before it arrives at the receiver. The dispersed signals on different paths may take different times to reach the destination. Thus, the resultant signal after summing up all dispersed signals may have been significantly distorted and attenuated when compared with the transmitted signal. The receiver may not recognize the signal and hence the transmitted data cannot be received. This unreliable nature of the wireless medium causes a substantial number of packet losses.

Mobility — Without the constraints imposed by the wired connections among devices, all devices in a wireless network are free to move. To support mobility, an ongoing connection should be kept alive as a user roams around.

In an infrastructured network, a handoff occurs when a mobile host moves from the coverage of a base station or access point to that of another one. A protocol is therefore required to ensure seamless transition during a handoff. This includes deciding when a handoff should occur and how data is routed during the handoff process. In some occasions, packets are lost during a handoff.

In an ad hoc network, the topology changes when a mobile host moves. This means that, for an ongoing data communication, the transmission route may need to be recomputed to cater for the topological changes. Since an ad hoc network may consist of a large number of mobile hosts, this imposes a significant challenge on the design of an effective and efficient routing protocol that can work well in an environment with frequent topological changes.

Limited Power and Energy — A mobile device is generally handy, small in size, and dedicated to perform a certain set of functions; its power source may not be able to deliver power as much as the one installed in a fixed device. When a device is allowed to move freely, it would generally be hard to receive a continuous supply of power. To conserve energy, a mobile device should be able to operate in an effective and efficient manner. To be specific, it should be able to transmit and receive in an intelligent manner so as to minimize the number of transmissions and receptions for certain communication operations. For instance, the total energy consumed for TCP is inversely proportional to its goodput [15]. An energy-efficient TCP should minimize the number of retransmissions.

PROBLEMS FOR TCP

The congestion control mechanisms of TCP have been designed with the assumption that all segment losses are congestive losses. Due to the specific characteristics of wireless networks described earlier, TCP suffers poor performance because of noncongestive segment loss (including random loss and burst loss) and packet reordering. These will be described in detail next.

Random Loss — The traditional congestion control measures for TCP has been designed for the wired network environment. The segment loss rate due to bit corruption and link errors is nearly negligible. In other words, almost all segment losses are congestive losses in wired networks. Indeed, the TCP congestion control mechanisms are generally reactive. When the loss of a data segment is inferred, network congestion is postulated. The size of the congestion window is reduced to assist in alleviating the congestion. Unfortunately, in a wireless network, the loss of a data segment does not necessarily correspond to network congestion because it may be dropped due to signal fading. It is typical to have a one percent to two percent random loss rate, say, for IS-95 code division multiple access (CDMA) based data service [16]. With the misinterpretation of the nature of segment loss, the congestion control mechanisms react inappropriately by keeping the sending rate of a TCP connection small and some data segments are retransmitted spuriously. This leads to inferior performance.

Burst Loss — A burst loss event may be initiated by signal fading. Prolonged uncontrollable channel interferences can lead to correlated packet losses. Yet, it generally occurs over a very short duration, leading to a loss of several consecutive segments at a time.

In an infrastructured network, all incoming and outgoing communications for a mobile host are routed via the base station it connects to. When it moves away from the coverage area of the base station, it needs to register at another base station in whose coverage area it moves. All subsequent communications are then routed via the new base station and the handoff process is completed. It can be shown [17] that the handover time for IEEE 802.11b wireless LANs typically takes one to two seconds to complete. However, a chain of packets delivered to a mobile host may be lost as they are routed to the old base station when a handoff is processed. Therefore, a handoff event can initiate a burst loss event. The frequency of the occurrence of a handoff event depends on the size of the coverage region and the mobility of the host involved.

The same situation can happen in an ad hoc network. Due to the mobility of some hosts, the network connectivity and hence the network topology may change. The network may sometimes even partition, say, for several seconds [10]. The transmission path for a traffic flow may be affected. The rerouting process for the traffic flow can take some time to complete. Thus, some packets belonging to the same traffic flow may be lost during the process. Thus, a burst loss event occurs whenever its transmission path is disrupted. Unlike the case for the infrastructured networks, the frequency of the occurrence of a burst loss event depends on the transmission range as well as the mobility of each host in the ad hoc network.

For either cases, a blackout due to mobility can lead to serial timer expirations for a connection so that multiple consecutive timer expirations and retransmissions of the same data segment take place within a single blackout period. The timeout period for the retransmission timer is doubled with each unsuccessful attempt until it reaches a value of at least 60 s [4]. Several consecutive retransmission failures can result in a terribly long period of inactivity of the connection even after the network conditions have been restored to normal.

Packet Reordering — Packet reordering refers to the network behavior where the receiving order of a flow of packets

[6] *Time division multiple access (TDMA) comprises all algorithms allocating time slots to transmission channels according to time division multiplexing (TDM) scheme.*

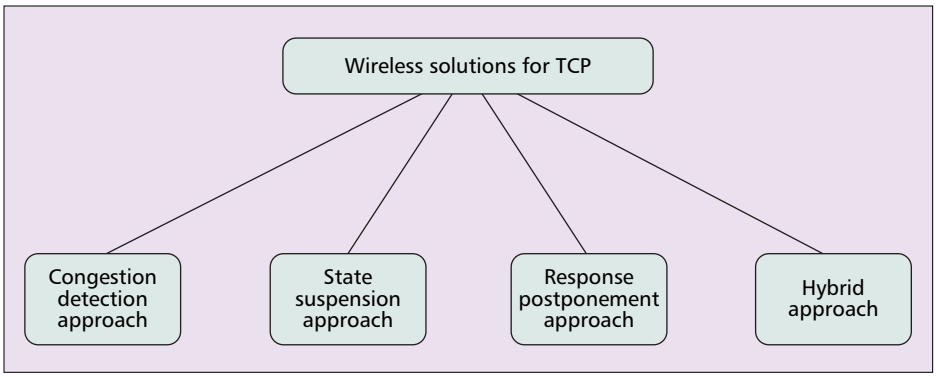

Figure 2. *The taxonomy of solutions for TCP in wireless networks.*

differs from its sending order. Recent studies [18, 19] show that packet reordering is not a rare event. The presence of persistent and substantial packet reordering violates the in-order or near in-order channel assumption made in the design of some traffic control mechanisms in TCP. It has been observed [18] that, in a half-second study of a 2.1 MB file transfer passing through MAE-East, nine out of ten fast retransmissions were triggered unnecessarily. This can result in a substantial degradation in application throughput and network performance [20].

In an infrastructured network, the occurrence of packet reordering is associated with a handoff event. When a mobile host is handed over from one base station to another, packets transmitted to or sent from the host are routed via the new base station instead of the old one after the handoff is finished. Since packets traveling on different paths may take different times to arrive at a destination, packet reordering can then happen.

In an ad hoc network, there is no fixed infrastructure and every mobile host can be a source, a destination, or a router. Topological changes in the network cause packets belonging to the same flow to be forwarded on different paths and arrive at a destination out of order.

In addition to the issues of mobility, link-layer retransmission is another cause for packet reordering. Several link-layer retransmission approaches [21, 22] have been proposed to recover transmission errors locally by retransmitting the lost frames at the link layer. Some schemes, such as the one proposed in [22], do not attempt to maintain in-order packet delivery for efficiency concerns since they are unaware of the semantics of the underlying transport protocol.

Taxonomy of Solutions for TCP in Wireless Networks

For a network environment where the noncongestive packet loss rate is negligible, an inferred segment loss is treated as a congestive loss. A source should retransmit the lost data segment and take appropriate congestion control actions to reduce the sending rate of a connection so as to avoid classical congestion collapse [23]. However, a general heterogeneous network environment can have both wired and wireless links. This means that, when a TCP flow passes through a wireless link, the noncongestive packet loss rate is no longer negligible. A TCP client should not blindly consider the loss of a data segment as an indication of network congestion and carry out the traditional congestion control measures. The source and the destination may have to gather additional information from the network to distinguish between these two major types of segment losses.

Furthermore, a network may reorder packets, in addition to dropping packets, when it supports link-layer retransmission and device mobility. Even when an arriving segment is a newly received segment and is not the expected one, a destination cannot determine whether the expected segment has been dropped in the network or is simply reordered. The source and the destination have to gather information from the network to differentiate between packet reordering and packet loss.

In the following section we survey the end-to-end solutions proposed to date with no scooping intermediaries for TCP in wireless networks. The discussed algorithms can be implemented in a TCP client to generate appropriate traffic control responses, and/or in a participating router to report its congestion status to a TCP client. The taxonomy used in our survey is depicted in Fig. 2. We categorize the solutions for TCP in wireless networks into four different strategies, namely, the congestion detection approach, the state suspension approach, the response postponement approach, and the hybrid approach. The congestion detection approach is a collection of methods that measure the current network conditions to determine whether network congestion has actually occurred and choose a proper traffic control strategy based on the measured information. In other words, it aims to perform proper traffic control by differentiating the congestive issues from the noncongestive ones. The state suspension approach represents a group of techniques that detects the current state of the network so as to decide when the communication activity of a TCP connection is suspended and when it can be resumed in order to avoid noncongestive losses. The state of the connection may or may not be readjusted based on the network conditions after the suspension. The response postponement approach is a class of solutions in which a TCP client delays triggering a traffic control response in order to alleviate the problems in wireless networks. The hybrid approach is a collection of methods that can be classified by more than one approach described above. Specifically, a TCP client may make use of a combination of mechanisms to collectively improve the TCP performance in wireless networks. Any solution that can be classified in the hybrid approach is not considered as a member of any other three approaches.

Overview of Existing Solutions

We describe some representative end-to-end solutions with no scooping intermediaries for TCP in wireless networks in this section. We classify and present these algorithms, compare them, and discuss their strengths and weaknesses. Readers can refer to Table 1 for an overview of the surveyed schemes.

Congestion Detection Approach

TCP-Peach/TCP-Peach+ — Akyildiz, Morabito, and Palazzo developed TCP-Peach [24] to deal with the adverse effects found in satellite networks with long propagation delays and high link error rates. TCP-Peach introduces two new algorithms, namely, sudden start and rapid recovery, to work with other traditional congestion control mechanisms [3], including congestion avoidance and fast retransmit. Dummy segments, which are low-priority segments with a copy of the recently transmitted data, are employed to probe for the availability of network resources. A successfully delivered dummy segment

Algorithms	Solution Approach	Types of Wireless Networks	Devices Involved			Additional Information Needed				Problems Solved		
			Source	Destination	Router	Timestamp	Priority Queue Mngmt.	Congestion Notification	Failure Notification	Random Loss	Burst Loss	Packet Reordering
ATCP	Hybrid	Ad Hoc	√		√			√	√	√	√	√
DelAck	Response Postponement	Ad Hoc		√								√
ELFN	State Suspension	Ad Hoc	√		√				√		√	
Freeze-TCP	State Suspension	Infrastructured		√							√	
ILC-TCP	State Suspension	Infrastructured	√								√	
JTCP	Congestion Detection	Unspecified	√	√		√				√		
TCP Veno	Congestion Detection	Unspecified	√							√		
TCP Westwood/ Westwood+	Congestion Detection	Unspecified	√							√		
TCP-ADA	Response Postponement	Ad Hoc		√								√
TCP-Casablanca	Congestion Detection	Unspecified	√	√	√		√			√		
TCP-DCR	Response Postponement	Unspecified	√									√
TCP-DOOR	State Suspension	Ad Hoc	√	√		√						√
TCP-Feedback	State Suspension	Ad Hoc	√		√				√		√	
TCP-Jersey	Congestion Detection	Unspecified	√	√	√			√		√		
TCP-Peach/ TCP-Peach+	Congestion Detection	Infrastructured	√	√	√		√			√		
TCP-Probing	Congestion Detection	Unspecified	√							√	A-TCP	

Table 1. *An overview of surveyed wireless solutions for TCP.*

indicates that unused network resources exist and the transmission rate can then be increased accordingly. A source utilizes some unused bits in the TCP header to label a dummy segment. When a destination receives a dummy segment, it acknowledges the receipt. It makes use of some unused bits in the TCP header of an ACK to indicate that the ACK is for the dummy segment. Once the source receives the ACK, it increments the value of *cwnd* by one if a counter, *wdsn*, is zero. Thus, *wdsn* controls whether the congestion window grows upon the receipt of an ACK for a dummy segment.

Sudden start, which substitutes slow start, aims to open up the congestion window faster. When sudden start is triggered, *wdsn* is set to zero. After a source sets *cwnd* to one and sends the first data segment, it transmits one dummy data segment for every $\{\tau/awnd\}$ until $(awnd - 1)$ dummy segments have been sent, where τ is the estimated RTT of the connection and *awnd* is the size of the advertised window in segments. Thus, *cwnd* can quickly be raised to the achievable value within one RTT. About one RTT after the last dummy segment has been transmitted, the ACKs of all successfully delivered dummy segments should have been received and congestion avoidance then begins.

Rapid recovery, which replaces fast recovery, aims to alleviate the performance degradation problem due to link errors. Like fast recovery, *cwnd* is halved in response to an inferred segment loss. *wdsn* is set to $\{w/2\}$, where w is the value of *cwnd* just before rapid recovery is triggered. When an ACK for a data segment arrives, a source sends two dummy segments until a total of w dummy segments have been transmitted. Upon receiving an ACK for a dummy segment, *wdsn* is decremented by one until *wdsn* becomes zero. To maintain ACK-clocking as fast recovery, *cwnd* is incremented by one when an ACK for a data segment or an ACK for a dummy segment, upon receiving $\{w/2\}$ ACKs for dummy segments, arrives. Once an ACK for the retransmitted segment is received, *cwnd* is reset to $\{w/2\}$ as in fast recovery. Rapid recovery is completed and congestion avoidance is then carried out. Every subsequent ACK arrival for a dummy segment will lead to an increment of *cwnd*. Thus, the size of the congestion window can be recovered once all dummy segments arrive successfully at the destination and their ACKs are delivered to the source.

A major merit for TCP-Peach is that it can maintain ACK-clocking when *cwnd* is smaller than the number of unacknowledged data segments, and w dummy segments can still be sent in the first $\{\tau/2\}$ after *cwnd* is halved. However, TCP-Peach has implicitly assumed that, if a congestive loss event happens, more than half of the dummy segments are lost in transit. In addition, all dummy segments can be successfully delivered to the destination if a noncongestive loss has occurred. The size of the congestion window can be reclaimed. Since dummy segments are sent at a rate doubled that before a loss event is conjectured, they may be dropped at the routers since such an increase in the traffic load can lead to congestion. This becomes more apparent when the connection RTT is large such that *cwnd* can be considerably large before a congestion loss can occur. Indeed, its performance improvement was substantially diminished when the packet loss rate exceeded about 5 percent [24]. Besides, there would be a wastage of network resources, since the delivery of dummy segments

does not result in any gain in connection goodput. Furthermore, all routers have to be configured to implement priority-based scheduling. Dummy segments and their ACKs are of lower priority than other regular segments and should be dropped first once network congestion develops.

Akyildiz, Zhang, and Fang extended TCP-Peach by proposing TCP-Peach+ [25] to further improve the network utilization. Instead of employing dummy segments, NIL segments carry unacknowledged data to allow a destination to recover the lost data segments. They also permit a sender to probe for the availability of network resources. TCP-Peach+ replaces sudden start and rapid recovery by jump start and quick recovery, respectively. Jump start is similar to sudden start, except that NIL segments are utilized instead of dummy segments.

To alleviate throughput degradation due to multiple segment losses within the same congestion window, the TCP selective acknowledgment (SACK) option [26] is applied to identify and assist in lost segment retransmissions. To avoid burst injection, a source cannot send more than a certain number of data segments at one time. Upon an ACK arrival during the quick recovery phase, a NIL segment can be sent only when no data segments are allowed to be transmitted. The source can send up to $\{w/2\}$ NIL segments for each invocation of quick recovery. The quick recovery phase terminates when an ACK acknowledges all data segments sent before the recovery phase is started.

TCP-Peach+ maintains ACK-clocking and it would not inject more data segments than before a recovery is initiated. Besides, it sends no more than one NIL segment for each ACK arrival. The probing overheads can then be reduced. Thus, TCP-Peach+ performed better than TCP-Peach as exhibited in [25]. However, as inherited from TCP-Peach, TCP-Peach+ still needs the participating routers to have differential treatments between regular segments and probing elements.

TCP-Probing — Lahanas and Tsaoussidis have proposed a sender-side solution, known as TCP-Probing [27]. It makes use of probing devices to determine whether network congestion has occurred when a segment loss is inferred. It aims to enhance the protocol performance against random loss and burst loss. TCP-Probing consists of two schemes, namely, adaptive TCP with probing (A-TCP) and selective probing (SP-TCP). A-TCP invokes a probing cycle upon the reception of three duplicate ACKs or a transmission timer expiration, in place of retransmitting the inferred loss segment as well as reducing the slow start threshold *ssthresh* and the size of the congestion window *cwnd*. During the probing cycle, probe segments are sent until the ACKs of a pair of probes are received within the specified time period.

The recovery process of A-TCP, which is diagrammed in Fig. 3, can be divided into three different cases. For the first case, both measured round-trip times (RTTs) of these probes are smaller than the best RTT.[7] A-TCP performs immediate recovery. When the probing cycle has started during the slow start phase, *ssthresh* and *cwnd* are set to 3/4 of the values prior to the detection of the loss. When the probing cycle is interrupted during the congestion avoidance phase, the size of the congestion window remains unchanged. For the second case, the measured RTT of the first probe is not smaller, and that of the second probe is smaller than the best RTT. Fast retransmit and fast recovery are performed as A-TCP considers the network congestion is relieving. For the default case, slow start is carried out.

Their results [27], obtained from an experimental testbed, showed that, when compared with two traditional TCP distributions, namely, TCP Tahoe [5] and TCP Reno [3], A-TCP yielded fewer timeouts and achieved more performance improvement with a higher packet error rate. However, a major drawback of A-TCP is that probing is costly to perform and responds slowly to noncongestive loss. Each probing cycle takes at least two RTTs to complete as it is triggered by an inferred loss event. When the packet loss rate becomes higher, A-TCP will initiate probing cycles more frequently regardless of the current congestion status of the network. Indeed, a normal data transmission is blocked during a probing cycle. Besides, the use of the probing RTTs and the best RTT may not be a reliable measure to determine whether network congestion has occurred in ad hoc wireless networks, since frequent route changes can lead to significant fluctuations in RTT of a connection and the best measured RTT is no longer a valid measure of network noncongestion.

SP-TCP was developed to reduce the rate of triggering a probing cycle. The main idea of SP-TCP is to avoid triggering more than one probing cycle within a certain small time interval during which the network congestion status is unlikely to change rapidly. To achieve this, a counter is maintained and incremented each time TCP estimates the RTT of the connection. Upon receiving three duplicate ACKs, a probing cycle is initiated only when the value of the counter is at least equal to the given threshold. Once a probing cycle is activated, the counter is then reset to zero. In case a probing cycle is skipped, fast retransmit and fast recovery are performed. Nevertheless, as in A-TCP, a probing cycle is triggered once whenever a retransmission timer expires, since it is inferred that network congestion has occurred in this case.

It has been shown [27] that SP-TCP could improve the TCP performance at low packet error rates, but it performed worse than A-TCP when the error rate becomes higher. When the packet error rate is high, it is more likely that multiple segments are dropped in the same congestion window. This explained [28] why TCP Reno implemented with fast retransmit and fast recovery was ineffective in handling multiple segment losses in the same window. Multiple segment retransmissions and reductions of the size and threshold of the congestion window before a timer expiration can result in a poorer performance of SP-TCP.

TCP Westwood/Westwood+ — Casetti *et al.* devised TCP Westwood (TCPW) [29], a sender-side solution for wired/wireless networks. TCPW adjusts the size of the congestion window upon an inferred segment loss by monitoring the rate of the acknowledged data. It can be considered as an extension from TCP Reno [3]. Traditionally, the congestion control mechanisms implemented in TCP halves the size of the congestion window upon the detection of a segment loss. However, the occurrence of a segment loss does not imply network congestion. This is especially true for wireless networks since wireless links are error-prone. Besides, the congestion window size is adjusted without the consideration of the current congestion level in the network. Such static, though reactive, congestion control approach cannot be effective for the general network scenario. Thus, TCPW, as its major merit, decouples congestion control from error control. The protocol performance becomes less sensitive to random packet loss due to lossy wireless links. Upon each ACK arrival, it uses the amount of new data acknowledged by that ACK to update the estimate for the available bandwidth of the connection. When

[7] *The best RTT is defined [27] as the minimum of the measured RTTs during the period between when the connection is initiated and when the current probing cycle is initiated. The probing RTTs are excluded for the determination of the best RTT.*

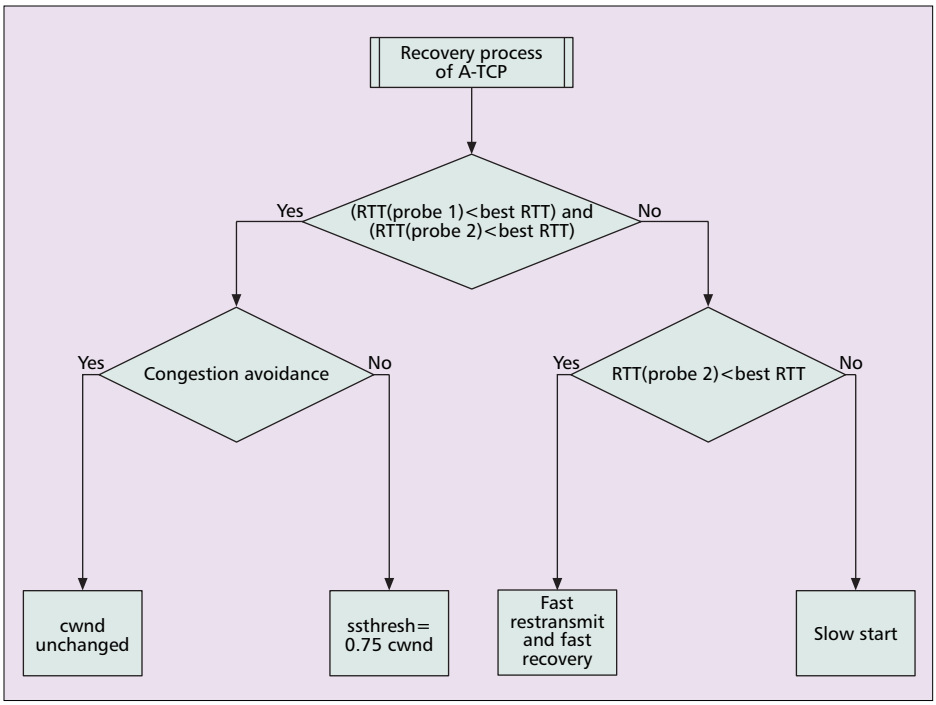

Figure 3. *The recovery process of A-TCP.*

a source TCP receives ACK n at Time t_n, the bandwidth sample taken from ACK n, b_n, can be computed as:

$$b_n = \frac{L_n}{t_n - t_{n-1}} \tag{1}$$

where L_n is the amount of data acknowledged by ACK n.

The estimated available bandwidth of the connection at Time t_n, $\hat{b}_n$, is obtained by applying a discrete-time low-pass filter obtained from the Tustin approximation as:

$$\hat{b}_n = \alpha_n \hat{b}_{n-1} + (1-\alpha_n) \cdot \frac{b_n + b_{n-1}}{2} \tag{2}$$

where

$$\alpha_n = \frac{2\xi - (t_n - t_{n-1})}{2\xi + (t_n - t_{n-1})}$$

and $\{1/\xi\}$ is the cutoff frequency of the filter. ξ was set to 0.5 s for the experiments shown in [29, 30]. To guarantee the frequency constraint specified by the Nyquist sampling theorem, a virtual null sample of $b_n = 0$ is introduced whenever a time period of $\{\xi/m\}$, where $m \geq 2$ has elapsed.

Slow start and fast retransmit are modified so that, instead of halving the value of *ssthresh*, *ssthresh* is assigned to the result of the product of the estimated available bandwidth and the minimum RTT sampled throughout the duration of the connection divided by the segment size.

The simulation results [29] exhibited that TCPW realized its throughput gain of 294 percent over than that of TCP Reno when the packet loss rate was 1 percent. It yielded 67 percent more throughput over TCP Reno when the mean blackout period was 0.1 s. It also achieved even better fairness when two connections with different RTTs are sharing the same channel. However, the performance gain of TCPW faded for either the packet loss rate was greater than about 0.5 percent or the blackout period was long, say, 0.5 s, since slow start was probably invoked more frequently to resolve burst loss due to a long blackout. In addition, TCPW demonstrated some unfriendliness to TCP Reno. The experimental results [29] showed that, while keeping the total number of connections sharing the same bottleneck link constant, the average throughput achieved by Reno connections dropped, though not to zero, with an increase in the number of TCPW connections when there was 1 percent packet loss on the link. Furthermore, TCPW may overstate the available bandwidth with the presence of ACK compression[8] [30].

Mascolo *et al.* devised TCP Westwood+ [30] to remedy the adverse effect of ACK compression in TCPW. To eliminate the high frequency components contained in the bandwidth samples due to ACK compression, a bandwidth sample is computed every RTT instead of with each ACK arrival. The value of m was taken to be four for generating the virtual samples. Their Internet experiments [30] revealed that TCP Westwood+ could estimate the available bandwidth more accurately than TCPW. It also inherited all other properties from TCPW.

TCP Veno — Fu and Liew introduced TCP Veno [31] with several sender-side refinements on top of TCP Reno [3] to deal with random loss in infrastructured wireless networks. TCP Veno estimates the backlog accumulated along the communication path of the connection. If the measured backlog is less than a certain threshold,[9] it is considered that the network does not experience congestion. An inferred segment loss is then regarded as a random loss. Otherwise, it indicates that network congestion has already occurred and hence any inferred segment loss is a congestive loss. All congestion control measures in TCP Reno are therefore adopted. It then utilizes the backlog information to define two congestion control enhancements. First, the additive increase algorithm is modified so that, upon the occurrence of network congestion, *cwnd* is increased by one every two RTTs instead of each RTT. This can be achieved by adding $\{1/cwnd\}$ to *cwnd* for every other new ACK received. A new ACK is an ACK acknowledging some previously unacknowledged data segment(s). Second, the multiplicative decrease algorithm is changed such that, when fast retransmit is triggered and a random loss is inferred, *ssthresh* is set to 0.8 *cwnd* in place of 0.5 *cwnd*.

TCP Veno adopts the same mechanism as TCP Vegas [32] to estimate the size of the backlog. A source uses the measured RTTs to compute the expected and actual rates. The expected rate is *cwnd* divided by the minimum measured RTT, whereas the actual rate is *cwnd* divided by the smoothed measured RTT. The backlog can then be estimated as the product of the minimum measured RTT and the difference

[8] *ACK compression is a phenomenon that ACKs arrive at a source closer together than they were sent by a destination. Their interpacket spacings have been altered due to network queuing.*

[9] *The backlog threshold of three segments was found to be a good setting [31].*

between the expected and actual rates. To deal with changing network conditions, TCP Veno resets the minimum measured RTT whenever a segment loss is detected.

Their experimental results [31] demonstrated that TCP Veno helped the connection to stay in its operating region longer. It improved connection throughput and experienced fewer timer expirations when the random loss rate was moderate (in the order of 0.01). Indeed, a wireless channel with IS-95 CDMA-based data service has a typical packet loss rate of around one percent to two percent with little correlation among losses. The fairness and friendliness have been examined in [31, 33]. There was only a small deviation in throughput among Veno connections. When Reno and Veno connections co-existed, a higher achieved throughput for Veno connections did not lead to a reduction in throughput for Reno connections when some Reno connections were replaced by Veno connections. It was attributed to the efficient utilization of the available bandwidth by TCP Veno. Nevertheless, TCP Veno has three shortcomings. First, the performance improvement realized by TCP Veno fades when the random packet loss rate is high (greater than a few percents). It keeps reducing the size of the congestion window each time a segment loss is detected. This occurs frequently with a high loss rate. Second, as inherited from TCP Reno, TCP Veno fails to satisfactorily deal with multiple segment losses in the same congestion window without resolving it through a retransmission timer expiration. This means that it can perform poorly with burst loss. Third, TCP Veno may not work well in ad hoc wireless networks since the backlog estimation is sensitive to the oscillation in RTT due to route change.

TCP-Jersey — Xu, Tian, and Ansari devised TCP-Jersey [34] as a router-assisted solution to differentiate noncongestive wireless segment loss from congestive segment loss and react accordingly. TCP-Jersey follows the same idea as TCPW [29] to observe the rate of data acknowledged by ACKs in order to estimate the available bandwidth for the connection, but its estimator is simpler. Upon receiving ACK n, the available bandwidth, B_n, is estimated as:

$$B_n = \frac{\tau B_{n-1} + L_n}{(t_n - t_{n-1}) + \tau} \tag{3}$$

where τ is the smoothed RTT, L_n is the amount of data acknowledged by ACK n, and t_n is the arrival time of ACK n.

Given the segment size S, the optimal size of the congestion window (in units of segments) upon the receipt of ACK n, $ownd_n$, is computed as:

$$ownd_n = \frac{\tau B_n}{S}. \tag{4}$$

TCP-Jersey adopts slow start, congestion avoidance, and fast recovery from TCP Reno [3], but it uses explicit retransmit instead of fast retransmit. Explicit retransmit simply performs a segment retransmission. The adjustment of the congestion window parameters is left to the rate control procedure, which sets *ssthresh* to *ownd*. *cwnd* is assigned to *ssthresh* when the connection is in the congestion avoidance phase.

To determine whether network congestion has occurred, TCP-Jersey makes use of a simple congestion notification scheme originated from explicit congestion notification (ECN) [35], known as congestion warning (CW). A router marks the congestion experienced (CE) bit in the Internet Protocol (IP) header of all packets when the average queue length exceeds a given threshold.

Upon receiving a packet with the CE bit set, the destination echoes the congestion information by setting the explicit congestion echo (ECE) bit in the header of all segments sent to the source until it receives a segment with the congestion window reduced (CWR) bit in the TCP header set by the source. Upon receiving an ACK, TCP-Jersey executes the available bandwidth estimation algorithm and computes the optimal size of the congestion window whenever they have not been run for one RTT. If the ACK is a new ACK without the congestion warning, TCP-Jersey proceeds as TCP Reno. Explicit retransmit and fast recovery are carried out when the source has just got a certain number of duplicate ACKs and the newly received duplicate ACK has no congestion warning. However, if the newly received ACK has activated the congestion warning, the rate control procedure is applied first. Then, the congestion control measures are carried out as in the case without the congestion warning.

The simulation results [34] revealed that TCP-Jersey achieved a much greater performance improvement in connection throughput than TCPW over a wireless link with random packet loss. It had an outstanding performance gain when the packet loss rate on a wireless link was at least 10 percent. It was also found that TCP-Jersey was fair in sharing the available bandwidth among connections of the same kind and exhibited a similar level of friendliness as TCPW. However, similar to TCP Reno, TCP-Jersey fails to deal with burst loss since it may have to resolve multiple segment losses in the same congestion window through a timer expiration, followed by halving *ssthresh* and slow start. Furthermore, all routers along the communication path have to be configured to be aware of the CW scheme. They can estimate the average queue length and mark the CE bit when the queue length exceeds the threshold. A destination is also required to interpret the CE and CWR bits so as to properly set the ECE bit in the header of a TCP segment sent to a source.

JTCP — Wu and Chen developed the jitter-based TCP (JTCP) [36] to deal with noncongestive loss in wireless networks. JTCP makes use of the jitter ratio as a loss ratio predictor to determine the congestion level of the network. The average jitter ratio is computed as the interarrival jitter for the most recent and the least recent segments in the congestion window divided by the difference in the receiving timestamps between the most recent and the least recent segments. The interarrival jitter for a pair of packets is the difference of the interpacket times at the receiver and sender between these two packets. When the average jitter ratio is less than $\{k/cwnd\}$, an inferred segment loss is regarded as a noncongestive loss. Otherwise, it is considered as a congestive loss. Here, k is a control parameter which should not be larger than *cwnd*, since the jitter ratio corresponds to k segments being queued in the network while a total of *cwnd* segments are injected into it.

Upon receiving three duplicate ACKs, fast recovery will be performed if an inferred congestive loss is detected and the preceding fast recovery has been carried out at least one RTT ago. Otherwise, immediate recovery will be performed instead. If a noncongestive loss is inferred, *ssthresh* is set to $D \cdot cwnd$, where D is a decrease factor which can take any value between 0.5 and 1, excluding 0.5.

When a retransmission timer expires, slow start will be applied if a loss event is considered as a congestive event. Otherwise, JTCP contemplates that a burst loss, which causes the timer expiration, may have occurred. Fast retransmit and fast recovery will be carried out for a noncongestive loss.

It has been shown [36] that JTCP outperformed TCP Reno [3], TCP Newreno [37], and TCPW [29] when the packet loss rate was at least 10 percent over a simulated wireless link,

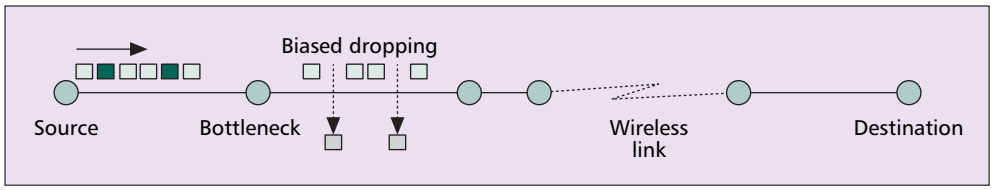

Figure 4. *Biased dropping for network congestion.*

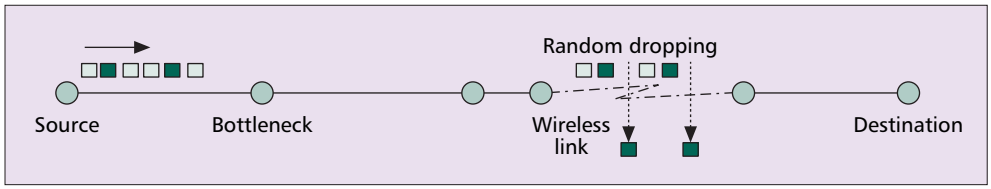

Figure 5. *Random dropping for wireless losses.*

where the values of k and D were taken as 1. This significant improvement in connection throughput was achieved by reducing unnecessary segment retransmissions and shrinking the congestion window size. Yet, there was no performance results for burst loss reported. The simulation results also demonstrated that JTCP was fair among competing flows, but its friendliness was not studied. Nevertheless, it is ineffective to resolve a burst loss event by fast recovery upon a retransmission timer expiration since multiple segment losses may have occurred within the same congestion window. As JTCP retransmits one segment for each timer expiration, this leads to the occurrence of several more timer expirations to recover the remaining segment losses in the same window, thereby causing substantial performance degradation. Besides, JTCP requires both the sender and receiver to insert and process timestamps in the TCP header.

TCP-Casablanca — Biaz and Vaidya proposed TCP-Casablanca [38], which applies a simple biased queue management scheme to discriminate congestion losses from random losses to improve the TCP performance over wireless networks. The key idea of TCP-Casablanca is to derandomize congestion losses so that the distribution of congestive losses differs from that of random wireless losses. When a source sends out a stream of new data segments to a destination, these segments are either marked "in" or "out" as their lost type such that one segment is labeled "out" for every k segments. In other words, if a segment is marked "out," the following $(k - 1)$ segments are marked "in." The marking pattern is then repeated for the next k new segments. Retransmitted segments are always marked "in." When a router experiences congestion, it drops those packets that have been labeled "out" before dropping "in" packets. By doing so, the dropping sequence will show correlated losses if the lost packets are dropped due to network congestion.

TCP-Casablanca uses a very simple function to determine whether the losses are due to congestion. The Casablanca loss discriminator function, $F(x, r, k)$, is defined as:

$$F(x,r,k) = 1 - \left\lfloor \frac{kx}{r} \right\rfloor \quad (5)$$

where x is the number of lost segments marked "out" and r is the number of losses in a measurement window of S ordered segments. If $F(x, r, k)$ is less than zero, the losses are diagnosed as congestive losses; otherwise, they are considered as noncongestive losses. The rationale is that most of the lost segments are marked "out" when network congestion occurs. Biased dropping occurs in a router with a bottleneck link, as illustrated in Fig. 4. If packets are dropped due to lossy wireless links, there will be no correlation between the loss type of a segment and the dropping probability. Packets are assumed to be dropped randomly on a wireless link, as exhibited in Fig. 5.

TCP-Casablanca has been extended from TCP Newreno [37], which is the same as TCP Reno [3] except that fast recovery exits only when a source receives an ACK acknowledging all data segments sent before it is entered. When a destination detects out-of-order segment arrivals, it computes $F(x, r, k)$ to determine the nature of the losses. If the losses are noncongestive, the destination marks the duplicate ACK with the explicit loss notification (ELN) before the ACK is sent to the source. Upon receiving three duplicate ACKs, the source does not halve the size of the congestion window. It carries out other traffic control operations as in TCP Newreno if the third duplicate ACK is labeled with ELN.

To eliminate the need of the receiver-side modifications, a sender-based TCP-Casablanca, known as TCP-Ifrane, has also been advocated in [38]. Upon the receipt of a duplicate ACK, a source checks whether the inferred loss segment has been marked "out." If it is so, the source simply classifies the loss as a congestive loss; otherwise, the loss is a noncongestive loss.

The simulation results [38] showed that TCP-Casablanca identified congestive losses with more than 95 percent accuracy and noncongestive losses with more than 75 percent accuracy. It outperformed TCP Newreno [37] and TCPW [29] by about 40 percent in connection throughput with the presence of 50 percent User Datagram Protocol (UDP) [39] traffic. However, TCP-Casablanca requires participating routers to have a differential packet dropping policy so that packets marked with "in" are dropped only when all queued packets labeled with "out" have been dropped. Besides, TCP-Casablanca does not perform well in the presence of other TCP-friendly flows when the congestive losses are dominant, because "out" segments belonging to Casablanca flows are dropped in advance of any other segments served by any router in the presence of congestion.

State Suspension Approach

Freeze-TCP — Goff *et al.* devised a receiver-side solution, known as Freeze-TCP [40], to improve the TCP performance in a network environment with frequent disconnections. A mobile host, which is a receiver of a TCP connection, continuously monitors the signal strengths of its wireless antennas and detects any impending handoffs. It sends some zero window advertisements (ZWAs) to force its peer, the sender of the connection, into the persist mode, whenever possible,

about one RTT before a handoff is expected to occur. A ZWA can be piggybacked into an ACK being transmitted to the source. When the source enters the persist mode, it freezes all retransmission timers and the size of the congestion window. Zero window probes (ZWPs) are sent, with their interprobe times being backed off exponentially, to the destination until it opens up its advertised window. When the destination responds to a ZWP with a positive advertised window size, the source exits from the persist mode and resumes its transmission as normal. To trigger a segment retransmission, the destination can also send three copies of the ACK for the last data segment received prior to the captioned disconnection.

However, Freeze-TCP has five major shortcomings. First, the network stack of a mobile host must be aware of mobility so that some cross-layer information exchanges, such as signal strengths of its wireless antennas, are needed. Second, a mobile host needs to predict when a disconnection is expected to happen in order to prevent the sender from transmitting segments to the mobile host before a disconnection occurs. Third, the scheme fails to predict and detect an upcoming disconnection event if it happens at a wireless link along the transmission path, where the end-points of the link are neither the sender nor the receiver of the connection. Thus, it can work fine in an infrastructured network where the core network is a wired network, but it does not function well in a multihop ad hoc network where a transmission path may consist of multiple wireless links. Fourth, the resumed transmission rate may be set inappropriately. There is no guarantee that the available bandwidth of a connection after a disconnection is more or less the same as that before it. If the available bandwidth is substantially reduced, the newly injected traffic can worsen the network congestion. On the other hand, a linear increase in the bandwidth consumption through congestion avoidance does not react quickly enough to probe for a significant increase in the available bandwidth. Fifth, the scheme can only avoid performance degradations due to disconnections. It fails to avoid and identify occasional segment losses because of signal fading.

ILC-TCP — Chinta, Helal, and Lee advocated an interlayer collaboration protocol for TCP in mobile and wireless networks, called ILC-TCP [41]. ILC-TCP is a sender-side solution to prevent performance deterioration due to temporary disconnections, where the sender is a mobile host. The basic idea is that a control decision for TCP upon a timer expiration is made based on the state information stored in the state manager. The state manager stores the state information exported from all the core layers of the network protocol stack to facilitate the collaboration and exchange of state information across these layers. Specifically, the link layer reports to the state manager the state (good or bad) of a link when it changes. A bad link state indicates that the channel fades or a handoff is imminent. Similarly, when a mobile host experiences an IP-level handoff, the corresponding handoff information is forwarded to the state manager. Once a handoff is completed, the state manager receives a notification indicating that the network layer connection is stable.

Upon a timer expiration, a source first checks with the state manager to determine whether both of its link layer and network layer connections are stable. If they are, it infers that network congestion has happened and the regular congestion control measures in TCP are then carried out. Otherwise, it considers that a temporary disconnection is impending and hence the state of the connection is frozen. When both the link layer and network layer connections become stable again, the connection is restored. The first unacknowledged data segment is retransmitted. The normal behavior of TCP is then followed.

ILC-TCP is similar to Freeze-TCP [40], except that ILC-TCP is a sender-side approach whereas Freeze-TCP is a receiver-based solution. Thus, both methods share their associated merits and limitations.

TCP-Feedback — Chandran *et al.* have proposed a router-assisted solution, TCP-Feedback (TCP-F) [42], to improve the performance of TCP for route failures in ad hoc wireless networks. Consider a communication path between a source mobile host and a destination mobile host going through a number of intermediate mobile hosts. A failure point is an intermediate mobile host which detects a route disruption due to the mobility of the next mobile host along the route. Once a route disruption is detected, the failure point transmits a route failure notification (RFN) packet to the source. Upon receiving the RFN packet, each intermediate mobile host invalidates the route. It also stops incoming packets to be forwarded through this route for the destination. If an alternate route exists, these packets can be rerouted through that alternate route and the RFN packet is discarded. Otherwise, it simply relays the RFN packet to the source.

When the source receives the RFN packet, it brings the TCP connection to the snooze state so that the communication activity for the connection is suspended. All the timers for the communication are marked as invalid and the state of the connection is frozen. A route failure timer is started to limit the maximum amount of time taken for the connection to be in the snooze state. The connection is reactivated upon either the route failure timeout or the reception of a route reestablishment notification (RRN) packet. When an intermediate mobile host which has forwarded an RFN packet to the source learns a new route to the destination, it sends an RRN packet to the source. The RRN packet for the same source-destination connection is transmitted once only by an intermediate mobile host so that all received subsequent RRN packets are discarded. Once the source receives the RRN packet, the connection is brought back to the active state. All unacknowledged segments are also retransmitted. The size of the congestion window is restored to that before the suspension.

Unlike Freeze-TCP, TCP-F is able to handle route disruption at any wireless link along the transmission path. However, TCP-F can inject traffic bursts to the network just after the communication is resumed, since it flushes out all unacknowledged segments upon connection reestablishment. This leads to the loss of ACK-clocking and far more bursty traffic, which may cause transient network congestion and congestion collapse from undelivered packets [23]. Similar to Freeze-TCP, TCP-F may set the sending rate inappropriately after the restoration of the connection since the available bandwidth of the connection may have changed. Moreover, while it can avoid performance degradations due to route interruptions, it fails to avoid and identify occasional segment losses because of signal fading.

ELFN — Holland and Vaidya made use of explicit link failure notification (ELFN) [43] to improve the TCP performance in mobile ad hoc networks. ELFN provides a source some information about link and route failures so as to avoid triggering congestion control measures for such failures. The idea of ELFN is similar to that of TCP-F [42]. Both techniques thus share their associated merits and limitations. However, they differ in the following two ways. First, ELFN relies on the route failures messages for dynamic source routing (DSR), instead of RFN packets in TCP-F, to notify a source about

link and route failures for a connection. Since source routing is assumed in ELFN, intermediate mobile hosts do not need to maintain or invalidate a route. The source disables the congestion control mechanisms and enters the "standby" mode in response to an ELFN notice. Second, ELFN does not require any intermediate hosts to send or forward a RRN packet to a source to reactivate a suspended connection. When the source is in the "standby" mode, it periodically probes the network to see if a route can be established for the connection. Upon receiving an ACK of a probe, the source restores its retransmission timers. The connection is then resumed as normal.

TCP-DOOR — Wang and Zhang developed TCP with detection of out-of-order and response (TCP-DOOR) [44] for mobile ad hoc networks. The out-of-order events are deemed to imply route changes in the networks, which happen frequently in mobile ad-hoc networks. The TCP packet sequence number and ACK duplication sequence number, or current timestamps, are inserted into each data and ACK segment, respectively, to detect reordered data and ACK packets. When out-of-order events are detected, a source can either temporarily disable congestion control or perform recovery during congestion avoidance. By temporarily disabling congestion control, the source will keep its state variable unchanged for a time period, say ξ_1 seconds, after detecting an out-of-order event. By instant recovery during congestion avoidance, the source recovers immediately to the state before the congestion response, which has been invoked within ξ_2 seconds ago.

However, TCP-DOOR may set the sending rate of a connection inappropriately after a route change. The available bandwidth of the connection may have varied significantly, but TCP-DOOR fails to probe for the change in the available bandwidth for the connection. Indeed, the state variables for congestion control are frozen for a time period. Besides, TCP-DOOR does not perform well in a congested network environment with substantial persistent packet reordering. It disables congestion control for a time period every time an out-of-order event is detected, which may lead to congestion collapse from undelivered packets [23].

Response Postponement Approach

DelAck — Altman and Jiménez have advocated the use of delayed ACK techniques, known as DelAck [45], to improve the TCP performance in multihop wireless ad hoc networks. DelAck is a receiver-side solution to reduce channel contentions among data segments and ACKs of the same TCP connection. It will also reduce performance degradation due to packet reordering. In a multihop ad hoc network, the forward and reverse traffic between two adjacent hosts on the transmission path for the same connection may share and contend for the same channel. The approach of this proposal is to delay acknowledging the arrivals of data segments and reduce the number of ACKs sent to a source. The connection overhead and hence the channel contentions can be reduced. The idea is to let a receiver generate an ACK for every d data segments. An ACK is also generated whenever the first unacknowledged data segment has been received for a certain time period, say 0.1 s. The value of d can be configured so that d increases with the segment sequence number.

However, there are two major drawbacks for this technique. First, the value of d is orthogonal to the segment sequence number in general. Indeed, the value of d may depend on the size of the congestion window, which in turn depends on the available bandwidth of a connection. Relating the value of d to the segment sequence number does not effectively reduce intra-flow channel contention, since the segment sequence number itself provides no information about the network congestion status. Second, bursts of TCP segments may be injected into the network every time a delayed ACK is received by a source. This can lead to transient network congestion and congestion collapse from undelivered packets [23].

TCP-ADA — Singh and Kankipati developed TCP with "adaptive delayed acknowledegment" (TCP-ADA) [46]. It is a receiver-side solution to reduce intra-flow channel contention in mobile ad hoc networks. The key idea of TCP-ADA is similar to that of DelAck [45]. However, DelAck defers acknowledgment until a certain number of data segments are received, while TCP-ADA postpones acknowledgment for a time period. Upon a data segment arrival, TCP-ADA updates Δ, an exponentially weighted moving average of the interarrival time between two successive segment arrivals. A destination will defer sending an ACK of the segment for a time period of $\beta\Delta$. The deferment period is restarted every time a data segment arrives before the deferment timer expires. An ACK is sent to a source if the total deferment period reaches a certain threshold. For the simulation experiments in [46], β and the maximum deferment period were set to 1.2 s and 0.5 s, respectively.

However, there are two major shortcomings for the method. First, if the data segments arrive at a destination so that they are spaced evenly, an ACK is sent after receiving a full congestion window of data. This means that a source has to be idle for about one RTT to receive an ACK before it can send new segments to the destination again. This results in the loss of ACK-clocking for the connection and injection of traffic bursts to the network. Second, there may be a significant drop in connection throughput if ACKs are dropped occasionally in the network. A destination may send just one ACK for a full congestion window of data. The loss of that ACK leads to a long idle period for the connection followed by the expiration of the retransmission timer and the initiation of the slow start phase to reopen the congestion window starting from one segment.

TCP-DCR — Bhandarkar *et al.* devised the delayed congestion response TCP (TCP-DCR) [47] to meliorate the TCP robustness to noncongestive events. The basic idea of TCP-DCR is to delay a congestion response for a time interval after the first duplicate ACK is received. The authors suggested to set this interval to one RTT so as to have ample time to deal with packet reordering due to link-layer retransmissions for loss recovery.

The simulation results in [47] demonstrated that TCP-DCR performed significantly better than TCP with SACK [26] and TCPW [29] in the presence of channel errors, where link-layer retransmissions are performed to recover from packet losses. Its performance improvement grew with the wireless delay. However, the chosen period of deferment is highly dependent on RTT. It does not take other network conditions affecting the extent of packet reordering into account. For example, it may not be a proper choice for a wireless channel with a high loss rate, since a lost packet may take a number of retransmissions before it can be sent successfully. In addition, a data segment may have been retransmitted on several wireless links in an ad hoc network before reaching the destination, so it can be delayed longer than one RTT. Further study is needed to find a proper choice of the delay interval in the presence of packet reordering.

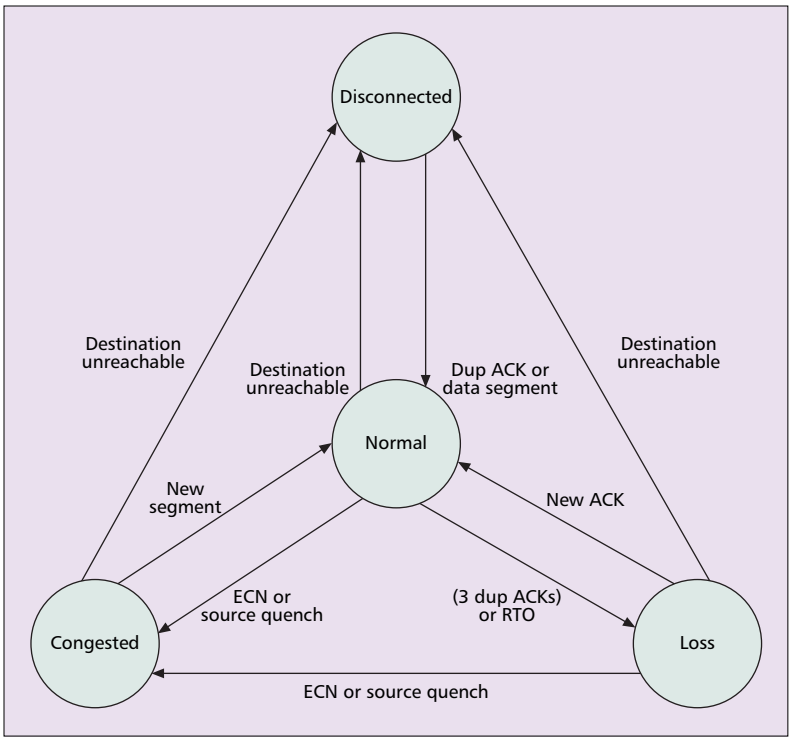

Figure 6. *The state transition diagram for ATCP.*

Hybrid Approach

ATCP — Liu and Singh proposed a sender-side, all-in-one solution, known as ATCP [10], to resolve the problems with TCP in ad hoc networks, such as high bit error rates, frequent route changes, network partitions, and packet reordering. The key idea of ATCP is to introduce a layer, called the ATCP layer, between TCP and IP at the sender's protocol stack so that the ATCP layer monitors the current TCP state and spoofs TCP from triggering its congestion control mechanisms inappropriately for problems specific to ad hoc networks. Besides, ATCP applies ECN [35] and Internet Control Message Protocol (ICMP) [48] to sense the onset of network congestion and the integrity of the transmission path.

The state transition diagram for ATCP is shown in Fig. 6. ATCP has four possible states, namely, normal, congested, loss, and disconnected. ATCP is in the normal state for a newly established TCP connection so that it does nothing and is transparent. When network congestion is experienced, a router sets the ECN flag when it processes a packet. Moreover, an ICMP source quench message can be sent to the sender directly. Once ATCP receives a message of either kind, ATCP transits to the congested state and does not interfere the congestion behavior of TCP. It returns to the normal state after a new segment is sent.

Whenever either three duplicate ACKs are received or the retransmission timer expires, it indicates that the transmission path between the sender and the receiver is lossy or some segments have been reordered in the network. In this case, ATCP puts TCP in the persist mode and enters the loss state. In addition, it sends the unacknowledged data segments from the TCP send buffer and maintains a separate set of timers to determine when these segments are retransmitted again if their ACKs are not received. It lets TCP leave the persist mode and goes back to the normal state when a new ACK is received.

ATCP goes to the disconnected state and places TCP into the persist mode when it receives an ICMP destination unreachable message for a packet transmission. This ICMP error message indicates that the transmission path is currently unstable because of mobility and/or network partition. The network will take some time before the network connectivity is re-established, if it has been partitioned, and a new route for the connection is computed. The source generates probes and sends them to the destination at exponentially increasing intervals up to the maximum period of 60 s. Upon receiving a duplicate ACK or a data segment, ATCP returns to the normal state. Since the available bandwidth for the connection after the blackout may be quite different with that before it, the transmission is resumed with the initial size of the congestion window being set to one segment. Slow start is invoked to search for the current available bandwidth.

As far as we know, ATCP is the only existing proposal that attempts to handle most of the problems relating to wireless networks. Since ATCP takes over the control for segment retransmissions whenever three duplicate ACKs are received or a retransmission timeout is going to occur, it successfully avoids taking any spurious congestion control measures which shrink the congestion window size unnecessarily. However, ATCP has three drawbacks. First, ATCP is inefficient in using the available bandwidth for data transmission in high-speed wireless networks with the presence of frequent route changes and network partition. Slow start is employed to probe for the available bandwidth after a blackout, but it is slow in growing the size of the congestion window when the available bandwidth or the RTT is large. Second, ATCP requires the mobile hosts in the ad hoc networks to be aware of and be implemented with ECN so that they can measure the average queue length and set the appropriate ECN flag when the queue length exceeds the given threshold. A destination is also required to interpret the ECN flag and forward the ECN information to a source. Third, ATCP does not allow a source to send new data segments to a destination when it is in the loss state as the source is in the persist mode. It can be in the loss state very often and new data segments are blocked from transmission. Hence, ACK-clocking cannot be maintained. This substantially degrades the TCP performance running in an error-prone wireless network. Further study is required to allow a source to continue sending new data segments in a lossy but noncongested wireless network whereas the congestion control mechanisms should not be invoked inappropriately.

Further Discussion and Open Research Issues

We summarize the properties of the presented algorithms in Table 1. These properties have already been discussed earlier. For the congestion detection approach, we find that the algorithms either use probes or the information stored in the data segments and ACKs to detect the congestion conditions of the networks. Specifically, TCP-Peach [24] and TCP-Peach+ [25] send low-priority segments to quickly seize the bandwidth available for the connection. TCP-Probing [27] transmits probes to detect whether the network is congested based on the estimated RTT. TCPW [29], TCP Westwood+ [30], TCP Veno [31], TCP-Jersey [34], and JTCP [36] estimate the congestion level of the network based on the spatial and temporal information carried by the ACKs. TCP-Casablanca [38] infers the network congestion status based on the ratio of the marked segments being dropped by the network.

These algorithms are generally able to distinguish between congestive loss and noncongestive random loss so as to determine the appropriate traffic control measure strategy to improve the TCP performance. Specifically, if a noncongestive event is inferred, the congestion control mechanisms are not activated so that the size of the congestion window can be sustained. However, most of these techniques are extended from TCP Reno [3] and hence they fail to gracefully handle multiple segment losses in the same congestion window. Besides, they provide no specific mechanisms to avoid burst loss due to temporary disconnections. They are therefore inadequate at alleviating performance problems due to burst losses and temporary disconnections resulting from mobility.

For the state suspension approach, we discover that the methods utilize the state information as well as route failure and restoration notifications to decide whether the communication activity of a connection is suspended or resumed. Freeze-TCP [40] uses the signal strength information to infer the occurrence of a temporary disconnection. ILC-TCP [41] freezes the communication whenever either link or network errors are experienced. TCP-Feedback [42] and ELFN [43] stop the communication activity when a route failure notification is received. They resume communication after a route is established for a suspended connection. TCP-DOOR [44] temporarily disables congestion control or performs instant recovery during congestion avoidance after detecting an out-of-order event. These algorithms, except TCP-DOOR, are successful at suspending any congestion control measures and stopping further segment losses when a temporary disconnection is encountered. However, they fail to deal with occasional random losses due to transient, short-lived link errors, say, resulted from signal fading. TCP-DOOR alleviates some performance problems caused by packet reordering, but it does not help to reduce bursty traffic and can exaggerate network congestion. Furthermore, TCP-DOOR does not provide any mechanisms to deal with noncongestive losses.

For the response postponement approach, the algorithms defer taking any traffic control measures to gather more network information to see if the decision needs to be changed. DelAck [45] and TCP-ADA [46] delay the issuance of an ACK so as to reduce the load of the control traffic and thus channel contention. They can also deal with packet reordering to a limited extent, although they are not designed to do so, as a result of delayed acknowledgment. TCP-DCR [47] postpones triggering the congestion control response to a newly received ACK. These solutions are able to reduce spurious retransmissions and thus maintain a larger congestion window with the presence of packet reordering, but they fail to clock out traffic during the deferment of a congestion response. Nevertheless, they provide no mechanisms to deal with noncongestive losses.

For the hybrid approach, we note that ATCP [10] uses the ECN information and source quench messages to detect the occurrence of network congestion. A loss is considered as noncongestive if the network is not congested. ATCP also utilizes the destination unreachable messages to detect temporary disconnections. The algorithm is able to resolve all three problems for wireless networks as stated earlier, except for the efficiency, heterogeneity, and ACK-clocking issues mentioned when it was presented.

Nevertheless, noncongestive segment losses and their retransmissions can lead to the loss of ACK-clocking and bursty traffic injection, but none of the existing algorithms has attempted to resolve the issues. In addition, none of them can rapidly and correctly probe for the available bandwidth of a connection after it is recovered from a temporary disconnection. Except for TCP-ADA, they have not considered how to optimize the traffic behavior of TCP in order to improve the network performance, say, by reducing channel contention. Furthermore, there is a lack of the theoretical basis to seek for an efficient and optimal way to handle the identified noncongestive losses. In summary, these issues are needed to be considered to resolve the problems for TCP in wireless networks.

We believe the following is desirable in a good TCP algorithm in wireless networks:

- Operate as a sender-based algorithm in order to yield high interoperability
- Achieve high connection throughput
- Have the ability to identify congestive and noncongestive segment losses accurately and rapidly
- Minimize the occurrence of segment losses
- Minimize spurious segment retransmissions and avoid retransmissions by timeouts
- Maintain ACK-clocking and avoid injecting traffic bursts into the networks
- Achieve low algorithm complexity

Although the existing algorithms provide some possible solutions to alleviate the problems of TCP in wireless networks, some issues have not been discussed in the literature and are potential research topics. They are listed as follows.

Integrated Solution for All Types of Wireless Problems

We have identified random loss, burst loss, and packet reordering as three major problems for TCP in wireless networks. Except for ATCP [10], none of the surveyed solutions can deal with all of the aforementioned problems. Upon receiving three duplicate ACKs or the retransmission timer about to expire, ATCP considers this as a signal of a wireless loss event. ATCP freezes the congestion window parameters, retransmits unacknowledged data segments, and stops transmitting new data segments until a new ACK arrives. With persistent packet reordering, many segment retransmissions may be performed spuriously. Not only does this block the regular segment transmissions and reduce the connection goodput, but this also shrinks the battery lifetime of a wireless host due to unnecessary retransmissions. Even so, it may also be undesirable to cease sending new data segments during the disruptive periods because of frequent noncongestive loss. Hence, further study is needed to devise an integrated solution for TCP that can solve all types of problems.

Improved Discriminator for Wireless Problems

The success of a solution to wireless problems hinges on how accurately it can correctly determine the type of the wireless problems encountered. If a problem is mistaken as another one, improper traffic control may exacerbate the problem. It has been noted [38] that it is much more costly to mistakenly identify a congestive loss as a noncongestive one than the other way round. However, there is a lack of a single discriminator that can correctly distinguish congestive loss, random loss, burst loss, and packet reordering in an efficient and effective manner. Further study is warranted to find such a discriminator.

Conclusions

In this article we have presented a comprehensive and in-depth survey of current research on running TCP in wireless networks. Infrastructured networks and ad hoc networks are

two major types of wireless networks. We have found that channel contention, signal fading, mobility, and limited power and energy are four major characteristics of wireless networks. With these distinguishing features of wireless networks, TCP suffers from random loss due to signal fading, burst loss from prolonged channel interference and temporary disconnection, and packet reordering because of rerouting and link-layer retransmission.

Existing algorithms were categorized into four different strategies, namely, the congestion detection approach, the state suspension approach, the response postponement approach, and the hybrid approach. The congestion detection approach is a collection of methods that measure the current network conditions to determine whether network congestion has actually occurred and choose a proper traffic control strategy based on the measured information. We found that the algorithms can generally distinguish between congestive loss and noncongestive random loss, allowing proper traffic control measures to be employed for TCP performance gain, but they fail to avoid burst loss and performance degradation due to temporary disconnections.

The state suspension approach represents a group of techniques that detects the current state of the network so as to decide when the communication activity of a TCP connection is suspended and when it can be resumed in order to avoid noncongestive losses. The algorithms are successful at suspending any congestion control measures and stopping further segment losses during a temporary disconnection, but they fail to deal with occasional random losses due to transient, short-lived link errors, say, resulting from signal fading.

The response postponement approach is a class of solutions in which a TCP client delays triggering a traffic control response in order to alleviate the problems in wireless networks. The algorithms are able to reduce spurious retransmissions and thus maintain a larger congestion window with the presence of packet reordering, but they fail to clock out traffic during the deferment of a congestion response and hence suffer performance degradation due to noncongestive loss.

The hybrid approach is a collection of methods characterized by more than one approach described above. The algorithm is able to handle random loss, burst loss, and packet reordering, but still suffers from the efficiency, heterogeneity, and ACK-clocking problems.

We also proposed some future research directions, including the need of a mechanism to resolve the potential loss of ACKclocking and bursty traffic injection, the investigation of a mechanism to rapidly and correctly probe for the available bandwidth of a connection after it is recovered from a temporary disconnection, the study of an efficient and optimal approach, with theoretical support, to handle the identified noncongestive losses, and the development of an integrated solution and the formulation of an improved discriminator for all types of wireless problems.

Acknowledgment

This research is supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project no. AoE/E-01/99). We would like to thank Milan Todorović for his participation during the initial phase of this research. We would also like to express our gratitude to the anonymous reviewers for their valuable comments and suggestions which assisted us in improving the quality of the article.

References

[1] J. Postel, "Transmission Control Protocol," Request for Comments, RFC 793, Protocol Specification, DARPA Internet Program, Sept. 1981.
[2] D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols," *ACM SIGCOMM Comp. Commun. Review*, vol. 18, no. 4, Aug. 1988, pp. 106–14.
[3] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," Request for Comments, RFC 2581, Network Working Group, Internet Engineering Task Force, Apr. 1999.
[4] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," Request for Comments, RFC 2988, Network Working Group, Internet Engineering Task Force, Nov. 2000.
[5] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 18, no. 4, Aug. 1988, pp. 314–29.
[6] M. S. Blumenthal and D. D. Clark, "Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World," *ACM Trans. Internet Tech.*, vol. 1, no. 1, Aug. 2001, pp. 70–109.
[7] K. Pentikousis, "TCP in Wired-Cum-Wireless Environments," *IEEE Commun. Surveys and Tutorials*, vol. 3, no. 4, 4th Quarter 2000, pp. 2–14.
[8] A. A. Hanbali, E. Altman, and P. Nain, "A Survey of TCP over Ad Hoc Networks," *IEEE Commun. Surveys and Tutorials*, vol. 7, no. 3, 3rd Quarter 2005, pp. 22–36.
[9] B. Sardar and D. Saha, "A Survey of TCP Enhancements for Last-Hop Wireless Networks," *IEEE Commun. Surveys and Tutorials*, vol. 8, no. 3, 3rd Quarter 2006, pp. 20–34.
[10] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks," *IEEE JSAC*, vol. 19, no. 7, July 2001, pp. 1300–15.
[11] V. O. K. Li and X. Qiu, "Personal Communication Systems (PCS)," *Proc. IEEE*, vol. 83, no. 9, Sept. 1995, pp. 1210–43.
[12] J. H. Schiller, *Mobile Communications, 2nd ed.*, Addison-Wesley, 2003.
[13] Y. Hu and V. O. K. Li, "Satellite-Based Internet: A Tutorial," *IEEE Commun. Mag.*, vol. 39, no. 3, Mar. 2001, pp. 154–62.
[14] A. Gupta, I. Wormsbecker, and C. Williamson, "Experimental Evaluation of TCP Performance in Multi-Hop Wireless Ad Hoc Networks," *Proc. IEEE MASCOTS 2004*, Volendam, The Netherlands, 4–8 Oct. 2004, pp. 3–11.
[15] H. Singh and S. Singh, "Energy Consumption of TCP Reno, Newreno, and SACK in Multi-Hop Wireless Networks," *ACM SIGMETRICS Perf. Evaluation Rev.*, vol. 30, no. 1, June 2002, pp. 206-216.
[16] D. Mitzel, "Overview of 2000 IAB Wireless Internetworking Wksp.," Request for Comments, RFC 3002, Network Working Group, Internet Engineering Task Force, Dec. 2000.
[17] H. Velayos and G. Karlsson, "Techniques to Reduce IEEE 802.11b MAC Layer Handover Time," *Proc. IEEE ICC 2004*, vol. 7, Paris, France, 20–24 June 2004, pp. 3844–48.
[18] J. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is Not Pathological Network Behavior," *IEEE/ACM Trans. Net.*, vol. 7, no. 6, Dec. 1999, pp. 789–98.
[19] V. Paxson, "End-to-End Internet Packet Dynamics," *IEEE/ACM Trans. Net.*, vol. 7, no. 3, June 1999, pp. 277–92.
[20] M. Laor and L. Gendel, "The Effect of Packet Reordering in a Backbone Link on Application Throughput," *IEEE Network*, vol. 16, no. 5, pp. 28–36, Sept./Oct. 2002.
[21] H. M. Chaskar, T. V. Lakshman, and U. Madhow, "TCP over Wireless with Link Level Error Control: Analysis and Design Methodology," *IEEE/ACM Trans. Net.*, vol. 7, no. 5, Oct. 1999, pp. 605–15.
[22] D. A. Eckhardt and P. Steenkiste, "A Trace-Based Evaluation of Adaptive Error Correction for a Wireless Local Area Network," *Mobile Networks and Applications*, vol. 4, no. 4, Dec. 1999, pp. 273–87.
[23] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Net.*, vol. 7, no. 4, Aug. 1999, pp. 458–72.
[24] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Trans. Net.*, vol. 9, no. 3, June 2001, pp. 307–21.
[25] I. F. Akyildiz, X. Zhang, and J. Fang, "TCP-Peach+: Enhancement of TCP-Peach for Satellite IP Networks," *IEEE Commun. Letters*, vol. 6, no. 7, July 2002, pp. 303–05.
[26] M. Mathis *et al.*, "TCP Selective Acknowledgment Options," Request for Comments, RFC 2018, Network Working Group,

Internet Engineering Task Force, October 1996.
[27] A. Lahanas and V. Tsaoussidis, "Improving TCP Performance over Networks with Wireless Components using 'Probing Devices," *Int'l. J. Commun. Systems*, vol. 15, no. 6, July-Aug. 2002, pp. 495–511.
[28] K. Fall and S. Floyd, "Simulation-Based Comparisons of Tahoe, Reno, and SACK TCP," *Comp. Commun. Review*, vol. 26, no. 3, July 1996, pp. 5–21.
[29] C. Casetti *et al.*, "TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks," *Wireless Networks*, vol. 8, no. 5, 2002, pp. 467–79.
[30] S. Mascolo *et al.*, "Performance Evaluation of Westwood+ TCP Congestion Control," *Performance Evaluation*, vol. 55, no. 1–2, Jan. 2004, pp. 93–111.
[31] C. P. Fu and S. C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks," *IEEE JSAC*, vol. 21, no. 2, Feb. 2003, pp. 216–28.
[32] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE JSAC*, vol. 13, no. 8, Oct. 1995, pp. 1465–80.
[33] Q. Pang *et al.*, "Performance Study of TCP Veno over WLAN and RED Router," *Proc. IEEE GLOBECOM 2003*, vol. 6, San Francisco, CA, USA, 1–5 Dec. 2003, pp. 3237–41.
[34] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for Wireless IP Communications," *IEEE JSAC*, vol. 22, no. 4, May 2004, pp. 747–56.
[35] K. Ramakrishnan and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," Request for Comments, RFC 2481, Network Working Group, Internet Engineering Task Force, Jan. 1999.
[36] E. H.-K. Wu and M.-Z. Chen, "JTCP: Jitter-Based TCP for Heterogeneous Wireless Networks," *IEEE JSAC*, vol. 22, no. 4, May 2004, pp. 757–66.
[37] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," Request for Comments, RFC 2582, Network Working Group, Internet Engineering Task Force, April 1999.
[38] S. Biaz and N. H. Vaidya, "De-Randomizing" Congestion Losses to Improve TCP Performance over Wired-Wireless Networks," *IEEE/ACM Trans. Net.*, vol. 13, no. 3, June 2005, pp. 596–608.
[39] J. Postel, "User Datagram Protocol," Request for Comments, RFC 768, Protocol Specification, DARPA Internet Program, 28 Aug. 1980.
[40] T. Goff *et al.*, "Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments," *Proc. IEEE INFOCOM 2000*, vol. 3, Tel Aviv, Israel, Mar. 2000, pp. 1537–45.
[41] M. Chinta, A. Helal, and C. Lee, "ILC-TCP: An Interlayer Collaboration Protocol for TCP Performance Improvement in Mobile and Wireless Environments," *Proc. IEEE WCNC 2003*, vol. 2, New Orleans, LA, Mar. 2003, pp. 1004–10.
[42] K. Chandran *et al.*, "A Feedback-Based Scheme for Improving TCP Performance in Ad Hoc Wireless Networks," *IEEE Pers. Commun.*, vol. 8, no. 1, Feb. 2001, pp. 34–39.
[43] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," *Wireless Networks*, vol. 8, no. 2–3, Mar. 2002, pp. 275–88.
[44] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-Of-Order Detection and Response," *Proc. ACM MOBIHOC 2002*, Lausanne, Switzerland, 9–11 June 2002, pp. 217–25.
[45] E. Altman and T. Jiménez, "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks," *Lecture Notes in Computer Science*, vol. 2775, Sept. 2003, pp. 237–50.
[46] A. K. Singh and K. Kankipati, "TCP-ADA: TCP with Adaptive Delayed Acknowledgement for Mobile Ad Hoc Networks," *Proc. IEEE WCNC 2004*, vol. 3, Atlanta, GA, USA, 21–25 Mar. 2004, pp. 1685–90.
[47] S. Bhandarkar *et al.*, "TCP-DCR: A Novel Protocol for Tolerating Wireless Channel Errors," *IEEE Trans. Mobile Computing*, vol. 4, no. 5, Sept./Oct. 2005, pp. 517–29.
[48] J. Postel, "Internet Control Message Protocol," Request for Comments, RFC 792, Protocol Specification, DARPA Internet Program, Sept. 1981.

BIOGRAPHIES

KA-CHEONG LEUNG [S'95, M'01] (kcleung@ieee.org) received a B.Eng. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 1994, an M.Sc. degree in electrical engineering (computer networks), and a Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, CA, in 1997 and 2000, respectively. He worked as Senior Research Engineer at Nokia Research Center, Nokia Inc., Irving, TX, from 2001 to 2002. He was Assistant Professor at the Department of Computer Science at Texas Tech University, Lubbock, TX, between 2002 and 2005. Since June 2005 he has been with the University of Hong Kong, Hong Kong, where he is Visiting Assistant Professor at the Department of Electrical and Electronic Engineering. His research interests include wireless packet scheduling, routing, congestion control, and quality of service guarantees in high-speed communication networks, content distribution, high-performance computing, and parallel applications. He is listed in the 60th (2006) edition of *Marquis Who's Who in America*.

VICTOR O. K. LI [S'80, M'81, SM'86, F'92] (vli@eee.hku.hk) received S.B., S.M., E.E., and Sc.D. degrees in electrical engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, MA, in 1977, 1979, 1980, and 1981, respectively. He joined the University of Southern California (USC), Los Angeles, CA, in February 1981, and became Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. Since September 1997 he has been with the University of Hong Kong, Hong Kong, where he is Chair Professor of Information Engineering at the Department of Electrical and Electronic Engineering. He also served as Managing Director of Versitech Ltd., the technology transfer and commercial arm of the University, and on various corporate boards. His research is in information technology, including all-optical networks, wireless networks, and Internet technologies and applications. He is a Principal Investigator of the Area of Excellence in Information Technology funded by the Hong Kong Government. Sought by government, industry, and academic organizations, he has lectured and consulted extensively around the world. He chaired the Computer Communications Technical Committee of the IEEE Communications Society 1987–1989, and the Los Angeles Chapter of the IEEE Information Theory Group 1983–1985. He co-founded the International Conference on Computer Communications and Networks (IC3N), and chaired its Steering Committee 1992–1997. He also chaired various international workshops and conferences, including, most recently, IEEE INFOCOM 2004 and IEEE HPSR 2005. He has served as an editor of *IEEE Network*, the *IEEE JSAC* Wireless Communications Series, and *Telecommunication Systems*. He also guest edited special issues of *IEEE JSAC*, *Computer Networks and ISDN Systems*, and *KICS/IEEE Journal of Communications and Networks*. He is now serving as an editor of *ACM/Springer Wireless Networks* and *IEEE Communications Surveys and Tutorials*. He has been appointed to the Hong Kong Information Infrastructure Advisory Committee by the Chief Executive of the Hong Kong Special Administrative Region (HKSAR). He is a part-time member of the Central Policy Unit of the Hong Kong Government. He also serves on the Innovation and Technology Fund (Electronics) Vetting Committee, the Small Entrepreneur Research Assistance Program Committee, the Engineering Panel of the Research Grants Council, and the Task Force for the Hong Kong Academic and Research Network (HARNET) Development Fund of the University Grants Committee. He was a Distinguished Lecturer at the University of California at San Diego, at the National Science Council of Taiwan, and at the California Polytechnic Institute. He has also delivered keynote speeches at many international conferences. He has received numerous awards, including, most recently, the Changjiang Chair Professorship at Tsinghua University from the Ministry of Education, China, UK Royal Academy of Engineering Senior Visiting Fellowship in Communications, the Outstanding Researcher Award of the University of Hong Kong, the Croucher Foundation Senior Research Fellowship, and the Order of the Bronze Bauhinia Star, Government of HKSAR, China. He is also a Fellow of the HKIE and the IAE.