TCP in MANETs – challenges and Solutions

Erlend Larsen

Norwegian Defence Research Establishment (FFI)

27 September 2012

FFI-rapport 2012/01289

1175

P: ISBN 978-82-464-2133-9
E: ISBN 978-82-464-2134-6

Keywords

Mobile Ad Hoc Nettverk

Metningskontroll

Transportprotokoll

Approved by

Torunn Øvreås	Project Manager
Anders Eggen	Director

English summary

Mobile Ad hoc NETworks (MANETs) have gained significant popularity through the last decade, not least due to the emergence of low cost technology and the pervasiveness of the IP protocol stack. The FFI-project 1175, ”Gjennomgående kommunikasjon for operative enheter”, is chartered with researching MANETs for use by the Norwegian operational military forces.

Self-organizing and self-healing wireless multihop networks, MANETs are aimed at supporting tactical domain communications with a high grade of mobility. These networks will interconnect with other networks in the Networking and Information Infrastructure (NII) using IP as the common connecting protocol.

The Transmission Control Protocol (TCP) is ”the protocol that saved the Internet”, most importantly because of its congestion control mechanism. It is a vital building stone in IP-based networks, but it faces serious challenges when used in MANETs, since MANETs are challenged with interference and high grade of mobility, from which wired networks are spared. Thus, to employ MANETs interconnected in the defense communication infrastructure, it is important to study the problems and the current state of the art of TCP in MANETs.

This report is aimed at introducing readers to the TCP protocol, describing the challenges that TCP faces in MANETs, and give an overview of ongoing research to adapt TCP to MANETs.

Sammendrag

Interessen for Mobile ad hoc nettverk (MANET) har økt betraktelig det siste tiåret, ikke minst på grunn av fremveksten av lavkost-teknologi og den store utbredelsen av IP-protokollstakken. FFI-prosjekt 1175, ”Gjennomgående kommunikasjon for operative enheter”, har i oppdrag å gjøre forskning på MANET for bruk av Norges operative militære styrker.

MANET, selvorganiserende og selvhelende trådløse multihopp nettverk, er rettet mot å støtte kommunikasjon i det taktiske domenet med en høy grad av mobilitet. Disse nettverkene vil være koblet sammen med andre nettverk i NII med bruk av IP som felles kommunikasjonsprotokoll.

TCP er kjent som ”protokollen som reddet Internett”, spesielt på grunn av dens mekanisme for metningskontroll. Den er en viktig byggestein i IP-baserte nettverk, men den står overfor alvorlige utfordringer når den brukes i MANET, ettersom MANET har utfordringer med forstyrrelser og høy grad av mobilitet, som kablede nettverk er spart for. Derfor, for å benytte MANET som en integrert del av Forsvarets kommunikasjonsinfrastruktur, er det viktig å studere utfordringene TCP har i MANET, og hva som er state-of-the-art på TCP for bruk i MANET.

Denne rapporten har som mål å gi lesere en introduksjon til TCP protokollen, å beskrive de utfordringene som TCP står overfor i MANET og gi en oversikt over pågående forskning for å tilpasse TCP til MANET.

Contents

1 Introduction

Military communication on a tactical level is becoming IP-based. This allows the employment of one common communication infrastructure for multiple systems, enabling the network-based defense paradigm. With IP-based connectivity comes also a desire to interconnect wired and wireless communications systems. There is an expectation that services used in wired networks today also will be available in the wireless domain.

In the wireless domain, cell-phone technology has shown users that IP-based web communication is feasible. However, this communication technology requires infrastructure in the form of a high-capacity backbone[1] network and one hop wireless communication between the client terminal and base stations that connect to the backbone.

Current wireless communication in the military tactical domain consists mainly of point-to-point radio links and one-hop broadcast voice/Situational Awareness (SA) data. However, there is a lot of ongoing work focusing on interconnecting the various radio systems using Mobile Ad hoc NETwork (MANET) technology, to create heterogeneous MANETs. MANETs are self-configuring infrastructure-less networks that adapt dynamically to changing environments. In contrast to cell-phone technology, MANETs are able to support multi-hop wireless communication over a shared medium. However, the capacity and performance of MANETs are much lower, compared to cell-phone networks, and informing future users and service developers on the limitations as well as the advantages of this technology is essential for proliferation of the MANET technology.

While MANET technology is very suitable for tactical communication, many IP-based protocols are not directly usable in MANETs. These protocols were developed in a strictly wire-based network domain, where attributes like interference and packet loss are less dominant and better controlled than in wireless multi-hop networks. For instance, queue loss is the sole contributor to packet loss, while medium-based bit errors are but non-existent. In MANETs, the Bit Error Rate (BER) is much higher than in wired networks (several orders of magnitude). Protocols that anticipate the cause of packet loss to be caused by queue tail drop may make the wrong assumption in MANETs, reacting badly in this situation.

The *Transmission Control Protocol (TCP)* has been, and continues to be, an essential protocol for Internet communication. Without its rate control, traffic congestion would have rendered the Internet useless. However, TCP makes several assumptions about the network. It assumes that network congestion, and not transmission errors, causes packet loss. It also assumes that the Round Trip Time (RTT) is relatively constant (little jitter) and that rerouting happens very quickly. None of these assumptions are easily satisfied in MANETs, which results in TCP having substantial problems when employed in such environments.

TCP has been improved several times after its first version in 1981. In recent years, the focus has mainly been on optimizations due to the ever-increasing link capacity of wired networks. These

[1] A backbone network with either wired links or dedicated point-to-point radio links.

proposals are not directly useful for the employment of TCP in MANETs. However, there has become more interest during the last decade in improving TCP for communication in multi-hop wireless networks, although the main TCP research activity still has been focused at High Performance Computing (HPC). The increased research effort on TCP in MANETs bodes well for a future where TCP and its excellent qualities is part of extending wire-based services into the wireless multi-hop domain.

This report focuses on the evolution and adaption of the TCP protocol. There are other protocols that aim to replace TCP entirely, for instance through implementing flow and congestion control on top of the User Datagram Protocol (UDP). These solutions are beyond the scope of this report.

The rest of the report is structured in the following way: In Chapter 2, TCP's origin and historical development in the wired domain is described. Chapter 3 presents the challenges of applying TCP to wireless multi-hop networks. An overview of proposed solutions for adapting TCP for use in MANETs is next presented in Chapter 4, and the report is concluded in Chapter 5.

2 TCP in wired networks

2.1 Introduction

TCP is a transport protocol that provides a number of services for higher layers in the OSI network architecture stack [1]. It guarantees that a stream of bytes sent from the sender program on one computer is delivered reliably and in the same order to the receiver program on the other computer. The counterpart to the reliable TCP service is the User Datagram Protocol (UDP), which provides a datagram service where latency is reduced at the cost of data delivery reliability. A few key features set TCP apart from UDP:

- Ordered data transfer.
- Retransmission of lost packets.
- Error-free data transfer.
- Flow control.
- Congestion control.

This report gives only a brief introduction to the functions of the TCP protocol, to give the reader an understanding of the basic functions in the TCP protocol, and the differences between different TCP variants. If more information is desired, this can be acquired through several sources, including the many RFCs (introduced in Chapter 2.3.1) that describe the functions of TCP formally, three innovative books covering TCP/IP by W. Richard Stevens [2, 3, 4], and also more easily digestive works such as P. D. Amer's presentation [5] or even Wikipedia.

2.2 Brief description of TCP

Network function: TCP is a transport layer protocol which hides the rigid IP layer restrictions of maximum packet length and potential packet delivery problems, and deliver a byte stream service where the application knows that all bytes sent to TCP will be delivered at the destination application in the correct order without packet loss. TCP handles retransmission of lost data, rearranges of out-of-order data, and helps minimize network congestion.

A vast number of applications utilize TCP. Among the most used are the World Wide Web (WWW), E-mail, File Transfer Protocol, Secure Shell, Peer-to-Peer (P2P)[2] file sharing. Even some streaming applications use TCP as the transmission protocol. TCP focuses on reliable delivery, and this may increase the delivery delay, since it must wait for retransmissions of lost messages or reorder out-of-order-messages. Thus, it is less suitable for traffic that requires low delay, e.g. interactive streaming, video conferencing and Voice over IP (VoIP).

The reliability of TCP depends on acknowledgment packets sent from the destination to the source, to confirm to the source that the destination has received the data. The source keeps track of each sent packet, and maintains a window for packets for which it awaits Acknowledgments (ACKs). A new packet is not sent until a slot in this window is available. In addition, a timer is kept from the time the packet was sent, in case a packet disappears or is corrupted. The packet is retransmitted if the timer expires.

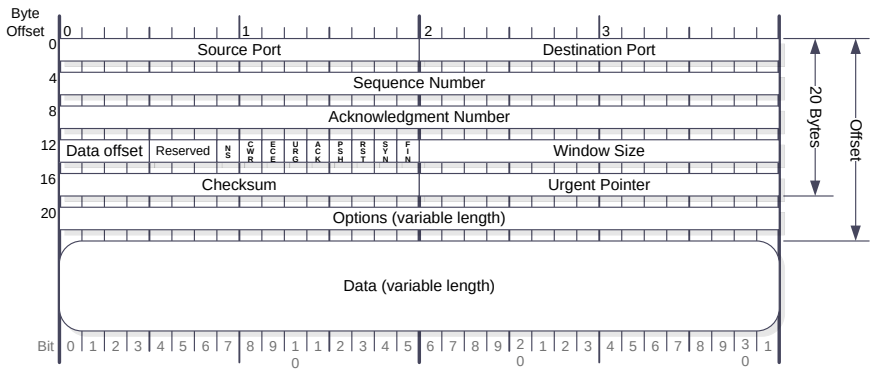

Figure 2.1 The TCP Packet Header.

TCP segment structure: TCP receives data from a data stream (from the application). The data are segmented into chunks and with an added TCP header, this accounts for a TCP segment. This TCP segment is transmitted over the network wrapped in an Internet Protocol (IP) datagram. The TCP header can be seen in Figure 2.1 and consists of the following fields:

Source port (16 bits) is the sender's port.
Destination port (16 bits) is the receiver's port.

[2]Due to fairness issues between multiple TCP flows, P2P solutions are currently researching better suited transport protocols, e.g. UDP with flow control.

Sequence number (32 bits) represents either (if the SYN bit is set) the initial sequence number[3], or (if the SYN bit is not set) the sequence number of the current packet. In the latter case, the sequence number of the first data byte will then be the initial sequence number plus one. If the SYN bit is not set, the sequence number is the position of the first byte of the current packet in the byte stream session plus the initial sequence number.

Acknowledgment number (32 bits) represents either (if the ACK bit is set) the sequence number of the next expected byte to be received from the sender (defined by the receiver)[4], or (if the ACK bit is not set – only occurs at the beginning of the communication) the acknowledgment of the other end's initial sequence number itself.

Data offset (4 bits) specifies the size of the TCP header in the number of 32 bit words, implicitly stating where in the TCP segment the data begins (the offset of the data in the TCP segment).

Reserved (3 bits) is for future use (set to zero).

NS (1 bit) Explicit Congestion Notification (ECN)-nonce concealment protection [6].

CWR (1 bit) If set, the Congestion Window Reduced (CWR) flag notifies the receiver that the sender has received an ECE flag and has reduced the congestion window as a result. [7].

ECE (1 bit) When the SYN bit is set, ECN-Echo (ECE) indicates whether the TCP peer is ECN capable. If the SYN bit is not set, a set ECE field indicates that a packet with the Congestion Experienced flag [7] in the IP header set is received during normal transmission.

URG (1 bit) indicates that the Urgent pointer field is valid.

ACK (1 bit) indicates that the Acknowledgment field is valid. (Should be set in all packets after the initial SYN packet sent by the client.)

PSH (1 bit) If set, requests the receiver to push the buffered data to the receiving application.

RST (1 bit) Connection reset.

SYN (1 bit) If set means that the sequence numbers should be synchronized between the sender and receiver. It is only set in the first packet sent from each end. Note that other flags' meaning depends on whether the SYN bit is set.

FIN (1 bit) If set means that the sender has finished sending data, and there will be no more data from the sender.

Window size (16 bits) is set by the receiver. Announces the maximum number of bytes (beyond the current ACK-ed sequence number) that the receiver is currently willing to receive.

Checksum (16 bits) is used for controlling if the header and data contain errors.

Urgent pointer (16 bits) represents an offset from the sequence number indicating the last urgent data byte, if the URG bit is set.

Options (0-320 bits) – Can contain various options, and a padding to ensure the field's length is divisible on 32 bit. For more information, refer to the following RFCs: [8, 9, 10].

The protocol operation of TCP can be seen as consisting of three phases. First, connections are established in a multi-step handshake process (*connection establishment*). Second, the *data transfer*

[3]The initial sequence number is randomly selected to avoid connection hijacking. Without randomly selecting the initial sequence number, this number would be easily guessable, allowing an attacker to blindly send a sequence of packets that the receiver would believe to come from a different IP address.

[4]The receiver thus acknowledges the receipt of all prior bytes (if any).

phase is entered. After the completion of the data transmission, the third phase is *connection termination* where the established virtual circuits are closed and all allocated resources are released.

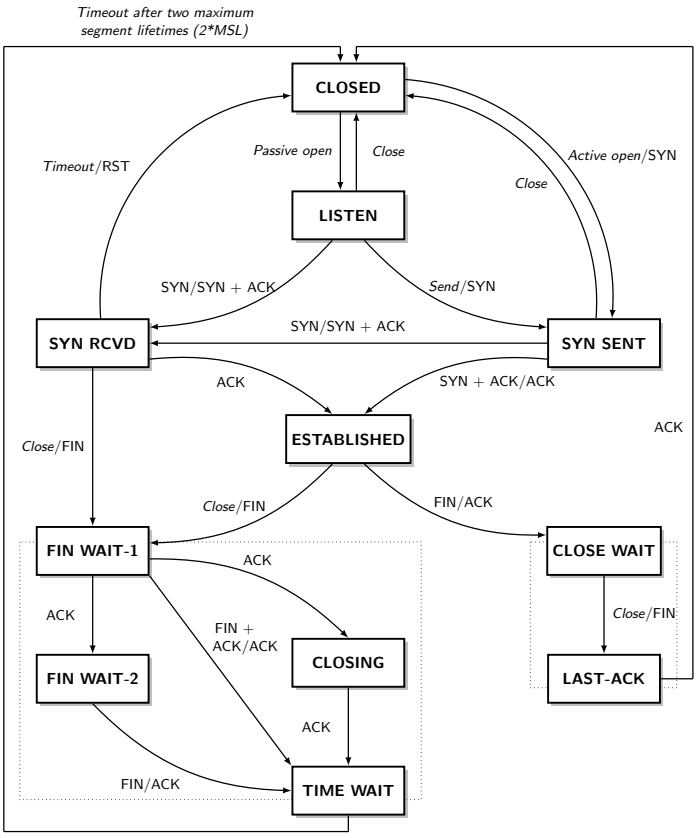

Figure 2.2 The TCP State-Transition Diagram, from [8].

A TCP connection is managed by an operating system through a programming interface, the Internet socket, which represents the local end-point for communications. The TCP connection has a relatively complex state machine (Figure 2.2). During the lifetime of a TCP connection it undergoes a series of state changes.

CLOSED The connection is closed.

LISTENING Waiting for a connection request from any remote client (only applies for server nodes).

SYN-SENT Waiting for the remote peer to send a TCP segment with the Synchronise (SYN) and ACK flags set. (usually set by TCP clients)

SYN-RECEIVED Waiting for the remote peer to send an acknowledgment after having sent back a connection acknowledgment to the remote peer. (usually set by TCP servers)

ESTABLISHED The port is ready to receive/send data from/to the remote peer.

FIN-WAIT-1 Waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2 Waiting for the server's Finish (FIN) segment. This indicates that the server's application process is ready to close and the server is ready to initiate its side of the connection

termination.

CLOSE-WAIT Waiting for a connection termination request from the local user.

LAST-ACK The server is in the process of sending its own FIN segment. The server's application process is ready to close and the server is ready to initiate its side of the connection termination.

TIME-WAIT Waiting for enough time to pass to be sure the remote peer received the acknowledgment of its connection termination request. According to [8] a connection can stay in TIME-WAIT for a maximum of four minutes, known as a Maximum Segment Lifetime (MSL).

Connection establishment: TCP uses a three-way handshake to establish a connection. The server must first bind to a port to open it for connections, before a client attempts to connect to it (called a passive open). Once the passive open is established, a client may initiate an active open, beginning the three-way handshake to establish a connection:

1. SYN: The active open is performed by the client sending a SYN to the server. It sets the segment's sequence number to a random value A.
2. SYN-ACK: In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number ($A + 1$). In addition, the server chooses a sequence number for communication in the opposite direction. This sequence number is another random number, B.
3. ACK: The client sends an ACK to the server. The sequence number is set to the received ACK value ($A + 1$), and the acknowledgement number is set to one more than the received sequence number, $B + 1$.

At this point, both the client and server have received an acknowledgment of the connection and the connection is established.

Flow control: TCP uses flow control to avoid having the sender send data too fast for the TCP receiver to receive and process it reliably. The flow control is managed using a sliding window mechanism. In the receive window field of each TCP segment, the receiver announces the amount of additional received data (in bytes) that it is willing to buffer for the connection. The sending host is not allowed to send more than that amount of data before it must wait for a window update from the receiving host.

The receiver may advertise a window size of 0. In such a case, the sender pauses sending data until a new advertised window of more than 0 is received. It could happen that the next window size update from the receiver is lost. Therefore, the sender starts the persist timer, which is used to protect TCP from a deadlock situation. The TCP sender will recover from a potential deadlock situation, when the persistence timer expires, by sending a small packet to the receiver so that the receiver can respond by sending an acknowledgement containing the new window size.

Congestion control is perhaps the most important aspect of TCP, which makes TCP capable of achieving high performance and avoid congestion collapse, at least in wired and single hop wireless networks. Where the flow control mechanism addresses the receiver's resources, congestion control addresses the network resources, preventing the sender to push too much traffic into the network.

Senders use the acknowledgments for data sent, and the lack of these, to infer network conditions between the sender and receiver.

The TCP congestion control algorithm has received much attention after its introduction in 1988, and a substantial number of proposals for improvement of the congestion control mechanism have been put forward. Most of these TCP variants, such as Tahoe, Reno, Vegas etc. have focused on congestion control, of which several are explained in Section 2.3.

Buffering small messages: TCP buffers outgoing messages that are smaller than one segment size for up to 200 ms. This behavior, known as Nagle's algorithm (see Section 2.3.1), is necessary to avoid very high overhead per payload byte. However, for some applications it can result in a very high delay, for instance for remote console applications like telnet, where TCP is used to communicate keystrokes. The message buffering is enabled by default, but the TCP header implements the Push (PSH) flag which signals to the protocol that the data should be forwarded to the receiver application immediately. In socket Application Programming Interface (API), the corresponding option is the TCP_NODELAY socket option.

Connection termination: The two sides of the connection perform the connection termination phase independently. When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which is acknowledged by the other end. A typical tear-down thus requires a pair of FIN and ACK segments from each TCP endpoint. After the conclusion of both FIN/ACK exchanges, the terminating side waits for a timeout before finally closing the connection. In this time span, the local port is unavailable for new connections. This prevents confusion if delayed packets of the current connection are delivered during subsequent connections.

A connection can be "half-open". In this case, one side has terminated its end, while the other has not. The side that has terminated can no longer send any data into the connection, even though the other side can. The connection can also be terminated by a three-way handshake, where host A sends a FIN, host B replies with a FIN & ACK (combining two steps into one), and host A replies with an ACK.

2.3 Historic development

2.3.1 The origins of TCP

The work on a transmission protocol for communication between "isolated" packet networks started as early as 1974 with the description of a TCP-like protocol by Vint Cerf and Bob Kahn [11]. In 1975, Ray Tomlinson introduced the three-way handshake [12]. The specification of TCP dates back to 1981 and the RFC 793 [8]. The specification has later been amended and changed, which has been documented through a large number of IETF RFCs [13, 9, 14, 15, 7, 16, 17], and a roadmap to the different documents specifying and extending TCP is presented by M. Duke et al. in [18].

Although several extensions and modifications of TCP have been proposed, most are changes to the sender side, leaving the protocol compatible with earlier versions. A milestone in the work on TCP

was New Year's Day 1983, when the Advanced Research Projects Agency Network (ARPANET) had officially completed its migration to the TCP/IP protocol suite.

In 1984, John Nagle proposed an algorithm now known as Nagle's algorithm [19]. The algorithm concatenates a number of small buffer messages. This increases the network efficiency through reducing the number of packets that must be sent. This again greatly reduces the overhead of small packets. The work predicted congestion collapse in the ARPANET.

The problems predicted by Nagle began occurring in October 1986, when the ARPANET saw several collapses caused by congestion. This spurred initiatives to address the problem. In 1987, Karn's algorithm [20] to better estimate the RTT in TCP was proposed, and in 1988, Van Jacobson and Michael Karels enforced TCP with congestion control. This was an extension to the existing flow control, which protected the receiver from being overrun. Today, the congestion control functionality has made TCP to be widely regarded as the protocol that "saved the Internet".

The congestion algorithm proposed by Jacobson and Karels opened a new field of research, focusing on the optimization of the congestion control mechanism. The next part of this report presents the most important TCP variants from the literature up until 1996 in chronological order.

2.3.2 TCP Tahoe

The first version of TCP with congestion control became known as *TCP Tahoe*[5] [22]. Tahoe was, in the same way as TCP Reno (Chapter 2.3.3), named after the variant of the 4.3 Berkeley Software Distribution (BSD) Operating System (OS) where they first appeared. These BSD OSs were themselves named after Lake Tahoe and the city of Reno, Nevada. The "Tahoe" algorithm first appeared in 4.3BSD-Tahoe (which was made to support the CCI Power $6/32$ "Tahoe" minicomputer), and was made available to non-AT&T[6] licensees as part of the "4.3BSD Networking Release 1"; this ensured its wide distribution and implementation.

The TCP Tahoe congestion control strategy consists of multiple mechanisms. For each connection, TCP maintains a *congestion window* that limits the total number of unacknowledged packets that may be in transit end-to-end. The congestion window is an extension of the sliding window that TCP uses for flow control. When a connection is initialized, and after a timeout, TCP uses a mechanism called *slow start* to increase the congestion window. It starts with a window of two times the Maximum Segment Size (MSS). Although the initial rate is low, the rate of increase is very rapid. For every packet acknowledged, the congestion window increases by one MSS so that effectively the congestion window doubles for every RTT. The window is doubled as follows: If the congestion window has two packets outstanding, and one packet is acknowledged, this means that the congestion window is increased to three packets, and only one packet is outstanding. I.e. the sender may now send two new packets. When the final packet (of the original two) is acknowledged, this allows the sender to increase the congestion window with one MSS yet again, bringing the total congestion window to

[5] The TCP nicknames for the algorithms appear to have originated in a 1996 paper [21] by Kevin Fall and Sally Floyd, which compares Tahoe, Reno, and SACK TCP using simulations.

[6] AT&T is no longer an acronym, but was originally an abbreviation for "American Telephone & Telegraph".

four, and of these two are free. In other words, the congestion window has doubled.

When the congestion window exceeds a threshold *ssthresh*, the algorithm enters a new state, called *congestion avoidance*. In some implementations (e.g., Linux), the initial ssthresh is large, resulting in the first slow start usually ending in a loss of a packet. The ssthresh is updated at the end of each slow start, and will often affect subsequent slow starts triggered by timeouts.

In the state of congestion avoidance, the congestion window is additively increased by one MSS every RTT, instead of the previous one MMS per acknowledged packet, as long as non-duplicate ACKs are received.

When a packet is lost, the likelihood of receiving duplicate ACKs is very high. (It is also possible, though unlikely, that the stream has undergone extreme packet reordering, which would also prompt duplicate ACKs.) Triple duplicate ACKs are interpreted in the same way as a timeout. In such a case, Tahoe performs a ”fast retransmit”, reduces the congestion window to one MSS, and resets to the *slow-start* state.

2.3.3 TCP Reno

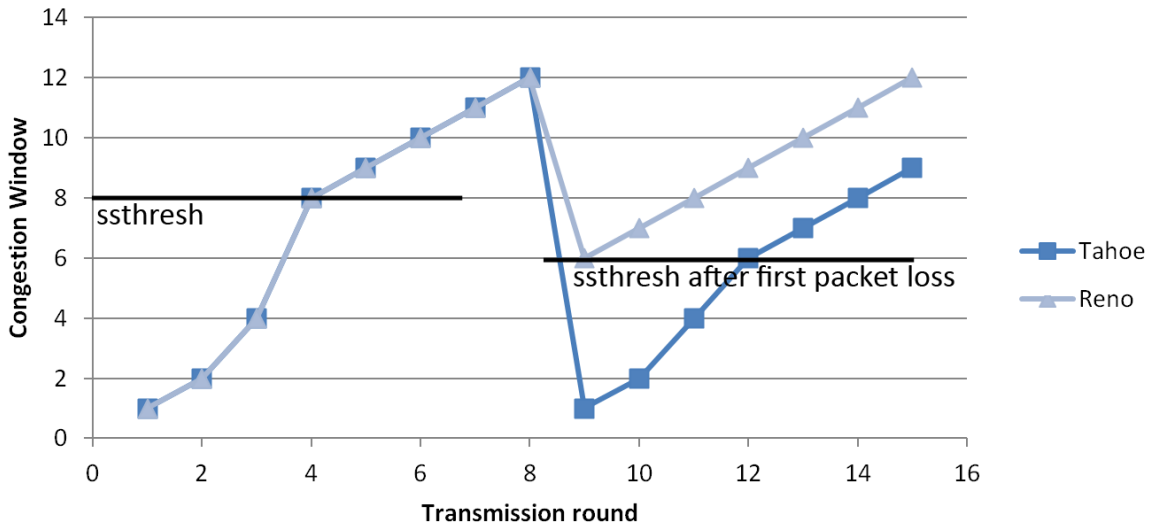

Figure 2.3 Congestion window development for Tahoe and Reno.

Improvements to Tahoe were made in 4.3BSD-Reno in 1990 and subsequently released to the public as ”Networking Release 2” and later 4.4BSD-Lite. The Reno version of TCP introduces a fast recovery phase. If three duplicate ACKs are received, Reno will halve the congestion window, perform a fast retransmit, and enter a state called *fast recovery*. In this state, TCP retransmits the missing packet that was signaled by three duplicate ACKs, and waits for an acknowledgment of the entire transmit window before returning to congestion avoidance. If there is no acknowledgment, i.e., if an ACK times out, TCP Reno experiences a timeout and enters the slow-start state, just like Tahoe. Figure 2.3 shows a comparison of the congestion window increase and the use of thresholds for Tahoe and Reno. Notice how the packet loss occurring after transmission round eight makes

Tahoe go into slow start and only beginning congestion avoidance in round 12, while Reno goes into fast recovery, whereby it halves its window and starts congestion avoidance directly.

2.3.4 RED and ECN

Random Early Detection (RED) [23] is an active queue management algorithm, as well as a congestion avoidance algorithm, proposed in 1993. In the traditional tail drop algorithm, a router buffers as many packets as it can, and simply drops the ones it cannot buffer. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows. Tail drop can also lead to TCP global synchronization as all TCP connections ”hold back” simultaneously, and then step forward simultaneously.

RED monitors the average queue size and drops packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped. RED is more fair than tail drop, in the sense that it is not biased against bursty traffic that uses only a small portion of the bandwidth. The more traffic a host transmits, the more likely it is that its packets are dropped, as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue. Early detection helps avoid TCP global synchronization.

Explicit Congestion Notification (ECN) [7, 24] is an extension to TCP dating back to 1994 that allows end-to-end notification of network congestion without dropping packets. It can be seen as an improvement of RED where packet drops are avoided, but it requires support by the TCP sender implementation. The extension is an optional feature that is only used when both endpoints support it and are willing to use it, and it depends on underlying network support to be effective. In the earlier variants of TCP, congestion is signaled by dropping packets. Using ECN, an ECN-aware router may set a mark in the IP header instead of dropping a packet in order to signal impending congestion. The receiver of the packet echoes the congestion indication to the sender, which must react as though a packet was dropped.

2.3.5 TCP Vegas

In 1993–1994, TCP Vegas [25, 26] was proposed. It provides a TCP congestion avoidance algorithm that uses packet delay, rather than packet loss, as a signal to help determine the rate at which to send packets. TCP Vegas detects congestion at an incipient stage based on increasing RTT values of the packets in the connection. Thereby, it can identify the queuing delay and based on this adjust the congestion window size. The difference between expected traffic and actual traffic is used to adjust the size of the congestion window. Both the increase and decrease of the rate is additive (Additive Increase, Additive Decrease (AIAD)).

The Vegas congestion detection algorithm differs from earlier TCP variants such as Tahoe and Reno, and also later variants like New-Reno (Chapter 2.3.7) and SACK (Chapter 2.3.8), where congestion

is detected by packet drops only after it has actually happened. Other TCP versions, such as Reno, keep increasing the sending rate until a packet is lost, and therefore they will *always* incur packet loss at some point or other. Vegas has achieved 40 to 70% better throughput than TCP Reno with less than half the packet loss [25]. In addition to its innovative congestion detection, Vegas still retains the default congestion detection mechanism, enabling packet loss detection through the default timeout if the other mechanisms fail.

In addition to the modified congestion avoidance mechanism, the TCP Vegas proposal also adapts the retransmission mechanism to avoid timeout if the sender never receives 3 duplicate ACKs (due to lost segments or window size is too small.). In such a case, the sender can do retransmission after one dupACK is received, if RTT estimate $>$ timeout.

Finally, the slow start phase is modified so that the sender tries to find the correct window size without causing a loss.

The Vegas algorithm depends heavily on accurate calculation of the base RTT value. If it is too small, then the throughput of the connection will be less than the bandwidth available, while if the value is too large, it will push too much traffic over the network path. Another challenge is the problem of rerouted paths, where the algorithm will have problems knowing the base RTT value. Finally, when TCP Vegas is run on a network that is running other variants of TCP that are less able to detect and act upon congestion, e.g., Reno, TCP Vegas will also get an unfairly small share of the bandwidth.

2.3.6 Improved startup behavior of TCP congestion control

In 1996, Janey Hoe proposed changes to the congestion control scheme in current TCP implementations to improve its behavior during the start-up period of a TCP connection [27]. The scheme uses acknowledgments from a receiver to dynamically calculate reasonable operating values for a sender's TCP parameters governing when and how much a sender can pump into the network. Since a TCP sender starts with default parameters, it often ends up sending too many packets and too fast during the startup period, leading to multiple losses of packets from the same window. Recovery from losses during this start-up period is often unnecessarily time-consuming, and the changes that Hoe proposed for the Fast Retransmit algorithm allow TCP to quickly recover from multiple packet losses without waiting unnecessarily for the timeout.

2.3.7 New-Reno

The New-Reno TCP variant was proposed in 1995–1996 by Floyd et al. [28]. It is a modification of TCP Reno, improving retransmissions during the *fast recovery* phase. In this phase, a new unsent packet from the end of the congestion window is sent for every duplicate ACK that is returned, to keep the transmit window full. For every ACK that makes partial progress in the sequence space, the sender assumes that the ACK points to a new hole, and the next packet beyond the acknowledged sequence number is sent. The progress in the transmit buffer resets the timeout timer, and this allows New-Reno to fill large or multiple holes in the sequence space. High throughput is maintained

during the hole-filling process, because New-Reno can send new packets at the end of the congestion window during fast recovery. When entering fast recovery, TCP records the highest outstanding unacknowledged packet sequence number. Upon the acknowledgment of this sequence number, TCP returns to the congestion avoidance state. New-Reno will misinterpret the situation if there are no losses, but instead reordering of packets by more than 3 packet sequence numbers. In such a case, New-Reno mistakenly enters fast recovery, but when the reordered packet is delivered, ACK sequence-number progress occurs and from there until the end of fast recovery, every bit of sequence-number progress produces a duplicate and needless retransmission that is immediately acknowledged. New-Reno substantially outperforms Reno at high error rates.

New-Reno is the default TCP variant for MS Windows XP.

2.3.8 SACK TCP

Selective Acknowledgment (SACK) TCP was another improvement to TCP proposed in 1996, in RFC 2018 [10]. The earlier variants of TCP, even back to Tahoe, implement a cumulative acknowledgment scheme where a lost packet results in duplicate ACKs for each subsequently received packet. Relying purely on the cumulative acknowledgment scheme can lead to inefficiencies when packets are lost. An example of this is a case where 10,000 bytes are sent in 10 different TCP packets, and the first packet is lost during transmission. Using the cumulative acknowledgment scheme, the receiver cannot say that it received the bytes 1,000 to 9,999 successfully, and only failed to receive the first packet, containing the bytes 0 to 999. Thus the sender may then have to resend all 10,000 bytes.

To solve the inefficient retransmission problem, TCP may employ the SACK option[7], which allows the receiver to acknowledge discontinuous blocks of packets that were received correctly. This is an additional mechanism to the sequence number of the last contiguous byte received successively, as in the basic TCP acknowledgment. The acknowledgment can specify a number of SACK blocks, where each SACK block is conveyed by the starting and ending sequence numbers of a contiguous range that the receiver correctly received. In the example above, the receiver would send SACK with sequence numbers 1,000 and 9,999, and the sender will therefore only retransmit the first packet, bytes 0 to 999.

The Selective Acknowledgment option is widely accepted, and has been enabled by default in Linux since kernel 2.2.

2.3.9 BIC & CUBIC

Binary Increase Congestion control (BIC)-TCP [29], from 2004, is an implementation of TCP with an optimized congestion control algorithm for high speed networks with high latency: so-called ”long fat networks” and has a unique congestion window algorithm. The algorithm tries to find the maximum where to keep the window at for a long period of time, by using a binary search algorithm.

[7] The SACK option is negotiated between the TCP endpoints, and is only used if support is advertised by both sides of a connection.

BIC-TCP was the default TCP variant for the Linux kernels 2.6.8 through 2.6.18.

In 2008, Ha et al. described and explained CUBIC[8] in [30]. CUBIC is a less aggressive and more systematic derivative of BIC-TCP. In CUBIC, the window is a cubic function of time since the last congestion event, with the inflection point set to the window prior to the event. CUBIC has been the default TCP variant for Linux since kernel 2.6.19 (2006), replacing BIC-TCP, but the CUBIC implementation has since gone through several upgrades. These are documented in [30].

2.3.10 Compound TCP

Compound TCP (CTCP) [31], proposed in 2005, is designed to aggressively adjust the sender's congestion window to optimize TCP for connections with large bandwidth-delay products while trying not to harm fairness. It is implemented as standard TCP version in Windows Server 2008, and is also available (but disabled by default) in Windows Vista and Windows 7.

The CTCP is claimed to be a synergy of delay-based and loss-based approach, where a scalable delay-based component is added into the standard TCP Reno congestion avoidance algorithm (a.k.a., the loss-based component). The sending rate of CTCP is controlled by both components. The new delay-based component can rapidly increase sending rate when the network path is underutilized, but gracefully retreat in a busy network when a bottleneck queue is built. The authors argue that augmented with this delay-based component, CTCP provides very good bandwidth scalability and at the same time achieves good TCP-fairness.

2.4 Usage in today's Internet

Medina, Allman and Floyd study the evolution of TCP variation usage in the Internet in [32], where they present usage numbers from February 2004. Some of the results are as follows:

- SACK is prevalent (in 2/3 of servers and 9/10 of clients).
- New-Reno is the predominant non-SACK loss recovery strategy.
- Duplicate Selective Acknowledgment (D-SACK) is gaining prevalence (supported by 40% of servers and at least 3% of clients).
- Most servers halve their congestion window correctly after a loss.
- Most web servers use packet counting to increase the congestion window.
- Most web servers use an Initial Congestion Window (ICW) of 1 or 2 segments.
- ECN is not common, with 93% classified directly as non-ECN-capable.
- The most widely used advertised window among clients is 64 KB with many clients using 8 KB and 16 KB, as well.
- Finally, most of the clients in the survey use an MSS of around 1460 bytes.

Yang et al. show in [33] that the use of congestion avoidance algorithm by the 5000 largest web servers in February 2011 was as follows:

[8] The name CUBIC is not an abbreviation, but rather a combination of the name of its originator, BIC, and the fact that it uses a cubic function to regulate the window growth.

- Only 16.85–25.58% of web servers still used the traditional Additive Increase, Multiplicative Decrease (AIMD).
- 14.36%, 15.82%, and 14.33% of web servers used BIC, CUBIC' (kernel 2.6.25 and before), and CUBIC (kernel 2.6.26 and after), respectively. Total = 44.51%.
- 9.97% and 0.30–9.03% of web servers use CTCP' (Windows Server 2003 and XP Pro x64) and CTCP (Windows Server 2008, Vista, and 7), respectively. Total = 10.27–19%.
- Surprisingly, some web servers use non-default TCP algorithms (such as Yet Another High-speed TCP (YeAH-TCP)), some web servers use some unknown TCP algorithms which are not available in any major operating system family, and some web servers use abnormal slow start algorithms.

The changes from 2004 to 2011 show that there is a move towards implementations of CTCP and CUBIC. These are TCP variations that are optimized for connections with large bandwidth-delay products. The challenges of MANET communications are thus not targeted by the major OS developers, and these are by default deciding the TCP variation selection for most users.

3 Challenges for TCP in MANETs

3.1 Introduction

The challenges for TCP in MANETs span all the layers below the transport layer in the OSI network stack [1]. At the PHY layer, interference and fading may result in bit errors and lost packets. At the MAC layer, the medium access may induce delay and is not able to totally avoid collisions, potentially causing packet loss if retransmission mechanisms are unable to salvage the problem. Retransmissions will also create delay and jitter. Some Medium Access Control (MAC) protocol implementations are able to dynamically change the data rate based on the transmission success. At the network layer, the routing protocol's delay in detecting topology changes may lead to periods without connectivity. Also, the end-to-end transmission time will change as a result of changing paths between the source and destination.

The IEEE 802.11 wireless stack [34, 35] is by far the most common wireless platform that is used for ad hoc networking today. Many MANET challenges have been identified based on work with the 802.11 platform, and in some cases the problems and subsequent solutions have focused more on mending the 802.11 standard than on addressing MANET problems generically. However, it is important to note that the 802.11 stack implements several mechanisms that are necessary for a functional MANET based on a Carrier Sense Multiple Access (CSMA)/Collision Avoidance (CA) MAC protocol. The functionality of 802.11 as a MANET carrier has been studied extensively, and in this TCP for MANET memo, the examination by Xu and Saadawi is particularly interesting. In 2001, Xu and Saadawi [36] examined how well – or rather how badly – TCP was supported in an IEEE 802.11 MAC MANET, with the focus on showing why the 802.11 protocol was unfit for MANET communication.

Mirhosseini and Torgheh give a good overview of the challenges of TCP in MANETs in [37]. In [38],

Papanastasiou et al. extensively evaluate several wire-based TCP variants (Reno, New-Reno, and Vegas) in different topology settings over the Ad hoc On-Demand Distance Vector Routing (AODV) [39] protocol. Their results reveal the performance merits of TCP Vegas and New-Reno in MANETs with respect to Reno, which is further explored and accounted for. Finally, the authors provide thorough insight into the behavior of TCP through extensive tracing on the interaction of TCP with the routing protocol.

The performance of several proposed MANET adaptations of TCP (TCP-F, ELFN, ATCP, Fixed RTO and TCP-DOOR) is investigated in [40].

3.2 The Physical layer

At the PHY layer, interference and fading may result in bit errors and lost packets. While wired links can now be regarded as so stable that one can ignore the probability of packet loss caused by bit errors, this is not the case with wireless links. For wireless links, the bit error rate is several orders of magnitude higher than wired links [41]. The TCP protocol was originally designed for wired networks, and its congestion avoidance mechanism does not consider link errors as a possible reason for packet errors or losses. Instead, TCP interprets packet losses caused by bit errors as congestion. This can significantly degrade the performance of TCP over wireless networks, when TCP unnecessarily invokes congestion control, causing reduction in throughput and link utilization.

3.3 The MAC layer

At the MAC layer, the contention based medium access may induce delay and is not able to completely avoid collisions, potentially causing packet loss if retransmission mechanisms are unable to salvage the problem. All MANET nodes share the same wireless medium. The contention and risk of collisions is much higher in such wireless networks than in the wired environment. The IEEE 802.11 is a CSMA/CA protocol, and work on such protocols [42] has shown that the TCP performance decreases drastically as the hop count is increased.

Retransmission mechanisms may also further increase the transmission delay, and create jitter as the number of needed retransmissions varies. A consequence of unsuccessful transmissions can also be a signal modulation change to improve the transmission success rate. This may result in a reduction of the bit rate. The IEEE 802.11 standard [35] states that if a node does not receive a link layer acknowledgement after retransmitting a DATA message 7 times (dot11ShortRetryLimit), the node must consider the link to be broken and should drop the DATA packet it tries to transmit.

It should also be noted that any MAC retransmission timeout must be kept at a significantly lower time frame compared to the retransmission timeout of TCP. If the two timeouts are too close, there is a chance that a packet may be retransmitted by TCP and by MAC at the same time, meaning that there will be duplicate TCP packets in the network, wasting resources.

Some MAC implementations, such as the IEEE 802.11, implements dynamic change of modulation to achieve the best performance in changing network conditions. For upper layer protocols, this may

lead to a high degree of variation in the available capacity. Another type of MAC layer capacity variation is Demand Assigned Multiple Access (DAMA), common in satellite communications, where the allocated bandwidth depends on the measured traffic load[9]. The allocated bandwidth may increase several times, creating unnecessary delay in achieving the desired and available capacity. In both cases, the underlying available capacity may vary, requiring the TCP protocol to adapt quickly and correctly in order to take full advantage of the available network resources.

Another problem with varying link technology over a path, both static and dynamic, is buffer bloat [43, 44]. Buffer bloat is the existence of excessively large and frequently full buffers inside the network, where they damage the fundamental congestion-avoidance algorithms of TCP. This problem is especially pronounced at bottleneck links.

TCP will, upon receiving bytes to transmit, wait up to 200 ms for more bytes to fill a MSS segment. The MSS size can be configured, but will usually default to the Ethernet Maximum Transmission Unit (MTU) minus the IP and TCP headers. In the case of the transfer of a large number of bytes, e.g. a file transfer, TCP will always produce MSS packets. This increases the risk of collisions in wireless networks.

In a heterogeneous MANET, a path used by TCP may consist of links with highly differing characteristics. The link types may vary from sat-com links with high delay, via stable, but low capacity Very High Frequency (VHF) links to unstable and short-range Ultra High Frequency (UHF) links. This will increase the challenges for TCP beyond the problems caused by one specific link technology. If one link technology could be anticipated, TCP could have been optimized for this, but with very diverse link technologies, link-specific optimizations will be difficult to implement.

3.4 The Network layer

At the network layer, the routing protocol's delay in detecting topology changes may lead to periods without connectivity and a risk of loops, both in case of mobility and fluctuating links. Also, the end-to-end transmission time/RTT[10] will change as a result of changing paths between the source and destination. If the RTT is increased too much, timeouts will occur on the TCP sender, causing unnecessary retransmissions.

If two neighboring nodes have different relative mobility, they will eventually become disconnected. Any routes using this link will fail, and it is the task of the routing protocol to detect the link break and discover an alternative route between the source and destination. In a MANET, this kind of topology change will happen on a fairly frequent basis, due to the limited communication range of radios.

Route failures and route changes may impact TCP in several ways. Route failures can cause packet drops at the intermediate nodes. These will be interpreted as congestion loss, a timeout event happens and TCP enters the slow-start process as if *congestion* occurred. Even if the routing protocol is able

[9] The current queue usage.

[10] The RTT is used by TCP to know the number of packets that are currently on their way to the destination.

to reroute the packets without packet loss, route changes can introduce frequent out-of-order packet delivery. The cumulative acknowledgement mechanism of TCP will generate duplicate ACKs before receiving the expected packet in sequence. If the sender receives three of such duplicate ACKs, TCP also presumes the network is congested and invokes fast retransmission.

3.5 The Transport layer

The TCP is an end-to-end protocol. It should be agnostic to the available performance and attributes of the lower layers. However, any solution that aims to improve TCP performance in MANETs by tuning the TCP protocol will have to deal with senders that may not be aware that the receiver, or part of the route, is in a MANET. As such, the end-to-end functionality of TCP is a challenge, since an interconnected MANET will enable connections between end-users that may have greatly differing versions of TCP implemented.

3.6 Cross-layer challenges

Cross-layer solutions are aimed at optimizing the network behavior across the layers of the network stack. However, as explained in [45], not all optimizations work well in all situations.

One example of unfortunate cross-layer behavior has even been identified while working on this report. It is not a direct TCP problem, but rather a problem with the use of Link Layer Notification (LLN) [46], which is a well-known cross-layer mechanism. Consider a network where a link is experiencing a high degree of bit errors and where a classic TCP variant like Reno or New-Reno is employed. The BER may be so high that although the MAC retransmission mechanism can handle most of the losses, sometimes a packet is lost. LLN is a mechanism to allow the routing protocol to discover link breaks immediately upon failure to receive MAC layer ACK after the maximum number of retransmissions. If the routing protocol considers the link down after one LLN and there are no other routes to the destination, the routing protocol will need to rediscover the same link before TCP packets can again be sent over it. In the meantime, the routing table will not contain an entry for the destination, and subsequent packets generated by the TCP sender are lost until the link is up again. However, the TCP congestion control will make sure that the interface queue is more or less filled with packets at the moment the LLN is received. Thus, the TCP sender will continue to transmit already routed packets over the link. The TCP receiver will in turn generate and transmit ACKs back towards the sender. Since a packet was lost, the TCP sender will transmit duplicate ACKs with the segment number of the last received packet before the lost packet. This activates a fast recovery phase at the TCP sender, but this retransmitted packet is not routed down to the interface, since the link is not considered as up by the routing protocol. Thus, the routing protocol and its use of LLN has broken the fast recovery phase of TCP, and when the link is rediscovered again (after a default time of 4 to 6 seconds), TCP will have to begin in slow start with its ICW of 1 or 2.

3.7 Challenges summary

TCP faces challenges at all lower layers in the network stack in MANETs, especially due to the congestion control mechanism which has problems differing between congestion and other network communication events. The effect is lower network utilization in a network which already struggles with low bandwidth. TCP might not be the best suited protocol in such environments.

On the other hand, MANETs are in great need of protocols that can adapt to changing bandwidths, providing flow and congestion control, and in-order packet delivery. The intended function of TCP makes it highly suited for MANETs.

The next section presents current proposals focused on the MANET challenges of TCP and solutions for these.

4 Solutions to improve TCP's performance in MANETs

4.1 Introduction

Several proposals to mend the many challenges encountered by TCP in MANETs have been generated through research. Some of these solutions are presented below. The intention is to give the reader an overview of the types of solutions that have been brought forward, for reference and for better understanding of the ways that TCP's challenges can be met.

IETF has addressed TCP's challenges through several RFCs. Two of them are RFC3135 and RFC3449. RFC3135 [47] is a survey from 2001 of Performance Enhancing Proxys (PEPs) employed to improve degraded TCP performance caused by characteristics of specific link environments. RFC3449 [48] presents best current practices (from 2002) with regards to network path asymmetry.

There are a number of surveys delving into the challenges of TCP in MANETs and possible solutions. Wang and Zhang presents a survey on TCP over MANETs, introducing three major challenges for TCP [49]. Two other publications that look at TCP and congestion control for MANETs are [50, 51]. Al Hanbali et al. present in [50] a survey of TCP alternatives for MANETs, classifying the alternatives in cross-layer and layered proposals. A very thorough survey of TCP and similar congestion control protocols for MANETs is presented in [51].

In the following subchapters, many solutions for TCP in MANETs are presented. The solutions are grouped according to the solution's requirement for changes, spanning from the Gateway (GW)-oriented low impact solutions that may be considered compatible with current TCP implementations, through solutions that changes the behavior of one or both the end-points, to solutions that require all nodes in the network to implement changes to support the solution:

- Gateway-oriented
- Changes limited to the source and/or the destination. These are further grouped according to compatibility:

  - TCP compatible solutions
  - TCP incompatible solutions
- Changes affecting relaying nodes

4.2 GW-oriented

GW-oriented solutions are solutions that require the entry point on the border of the MANET to have special functions, in order for the solution to perform as desired.

DSProxy: In [52], Skjervold et al. present a Service Oriented Architecture (SOA) approach, proposing a novel prototype proxy solution which adds both delay and disruption tolerance to Simple Object Access Protocol (SOAP). The ”Delay and disruption tolerant SOAP Proxy” solution can bridge heterogeneous networks and offers store-and-forward capabilities, delay tolerant network capabilities and swappable transport protocols while retaining backward compatibility with Commercial Off-The-Shelf (COTS) Web service clients and servers. The proxy solution does not rely upon parsing or inspecting the SOAP messages, which allows for end-to-end security through encryption.

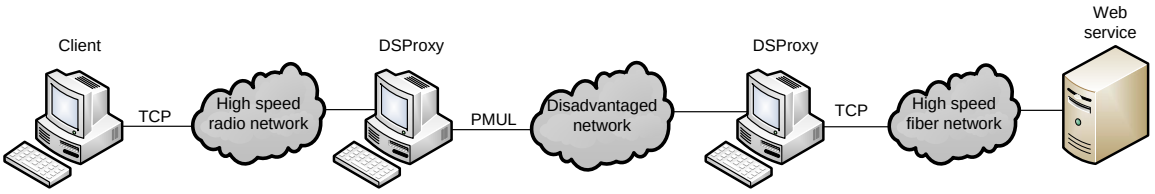

Figure 4.1 Proposed use of DSProxy, from [52].

The authors focus on the main challenge of using Web services in tactical communication systems, with low bandwidth, high error rates and frequent disruptions. Web services are commonly used with XML over TCP. Therefore, the authors propose to split the TCP stream (Figure 4.1) using a node that terminates the TCP flow and forwards the data over the disadvantaged grid using a better suited protocol such as UDP or A protocol for reliable multicast messaging in bandwidth constrained and delayed acknowledgement (EMCON) environments (PMUL). PMUL is a multicast protocol for Emission Control (EMCON) environments [53].

Split-TCP [54] is a solution for TCP seeking to resolve the unfairness suffered by connections with a large number of hops, compared to connections with a low number of hops. The scheme separates the functionalities of TCP congestion control and reliable packet delivery. For any TCP connection, certain nodes along the route take up the role of being proxies for that connection (Figure 4.2). The proxies buffer packets upon receipt and administer rate control. The buffering enables dropped packets to be recovered from the most recent proxy. The rate control helps in controlling congestion on inter-proxy segments. Thus, by introducing proxies, shorter TCP connections are emulated, and better parallelism in the network is achieved. The simulations show that the use of proxies abates the problems described as follows:

a) it improves the total throughput by as much as 30% in typical scenarios.
b) it reduces unfairness significantly. In terms of an unfairness metric that is introduced, the

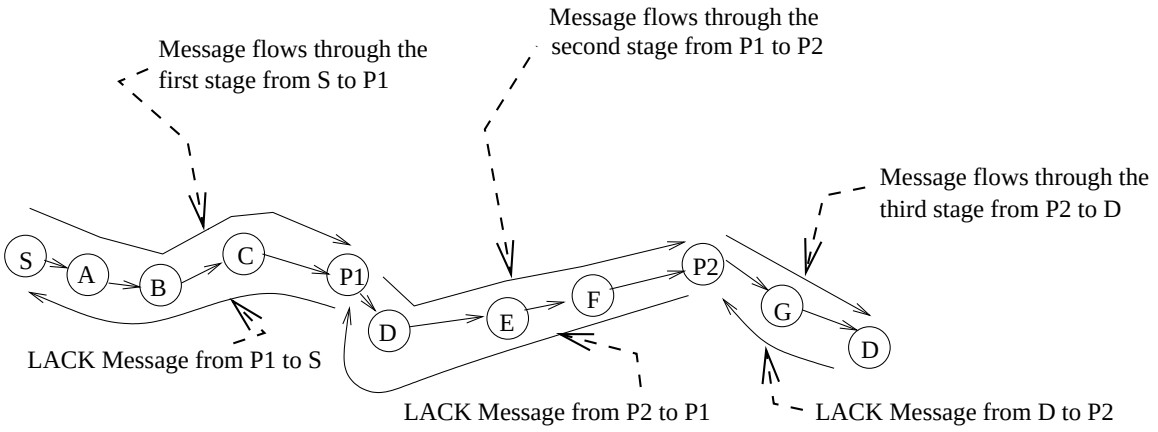

Figure 4.2 Split-TCP: TCP with proxies, from [54].

unfairness decreases from 0.8 to 0.2 (1.0 being the maximum unfairness).

The authors conclude that incorporating TCP proxies is beneficial in terms of improving TCP performance in ad hoc networks.

TCP Gateway Adaptive Pacing (TCP-GAP): In [55], the authors introduce an effective congestion control pacing scheme for TCP over multi-hop wireless networks with Internet connectivity. The pacing scheme is implemented at the wireless TCP sender as well as at the Internet gateway, and reacts according to the direction of TCP flows running across the wireless network and the Internet. The authors analyze the causes for the unfairness of oncoming TCP flows and propose a scheme to throttle aggressive wired-to-wireless TCP flows at the Internet gateway to achieve nearly optimal fairness. The proposed scheme, which is denoted as TCP-GAP, does not impose any control traffic overhead for achieving fairness among active TCP flows and can be incrementally deployed since it does not require any modifications of TCP in the wired part of the network. The authors show, in an extensive set of experiments using ns-2, that TCP-GAP is highly responsive to varying traffic conditions, provides nearly optimal fairness in all scenarios and achieves up to 42% more goodput for FTP-like traffic as well as up to 70% more goodput for HTTP-like traffic than TCP New-Reno. The sensitivity of the considered TCP variants to different bandwidths of the wired and wireless links with respect to both aggregate goodput and fairness is also investigated.

4.3 Changes limited to the source and/or the destination

Here, solutions that would impact only the source or the destination if implemented, are presented. These are further divided into two sub-groups, depending on whether or not they are compatible with the existing standard TCP implementations.

4.3.1 TCP compatible solutions

Ad Hoc TCP (ADHOCTCP): Mirhosseini and Torgheh [37] propose to improve TCP for MANETs through a solution named ADHOCTCP, by identifying three packet loss inducing network states and

have TCP act according to the packet loss reason. The three states are *Congestion*, *Channel error* and *Disconnection*. The congestion state is identified by use of two end-to-end measured metrics, Inter-packet Delay Difference (IDD) and Short Term Throughput (STT), which complement each other. A situation where IDD is high and STT is low is defined as a congested state. The disconnection state is identified using Explicit Packet Loss Notification (EPLN) which is based on the Dynamic Source Routing (DSR) protocol and an Internet Control Message Protocol (ICMP) ”Host unreachable” message. Finally, the channel error state is assumed after Retransmission Timeout (RTO) if the network state is not detected as congestion by the end-to-end measurements.

Ad Hoc TCP (ADTCP) [56] performs multi-metric joint identification for packet and connection behaviors based on end-to-end measurements, to robustly detect network states in the presence of measurement noise. The metrics, measured at the transport layer, are Inter-packet Delay Difference (IDD), Short Term Throughput (STT), Packet out-of-order delivery ratio (POR) and Packet loss ratio (PLR). The solution relies solely on end-to-end mechanisms. Using the technique, a network event is acted upon only if all the relevant metrics detect it.

Dynamic congestion window limit: In [57], Chen et al. address how to properly set TCP's Congestion Window Limit (CWL) to achieve optimal performance. The authors turn the problem of setting TCP's optimal CWL into identifying the Bandwidth-Delay Product (BDP) of a path in a MANET. They first show and prove that, independent of the MAC layer protocol being used, the BDP of a path in MANET cannot exceed the Round-Trip Hop-Count (RTHC) of the path. Further, the upper bound is refined based on the IEEE 802.11 MAC layer protocol, and show that in a chain topology, a tighter upper bound exists which is approximately $\frac{1}{5}$ of the RTHC of the path. Based on this tighter bound, the authors propose an adaptive CWL setting strategy to dynamically adjust TCP's CWL according to the current RTHC of its path. Simulations show that the strategy improves TCP performance by 8% to 16% in a dynamic MANET environment.

Dynamically delayed ACK: TCP performance over a static multi-hop network that uses the IEEE 802.11 protocol for access is studied by Altman and Jiménez in [58]. For such networks, the TCP performance is mainly determined by the hidden terminal effects (and not by drop probabilities at buffers) which limits the number of packets that can be transmitted simultaneously in the network.

The authors propose new approaches for improving the performance based on thinning the ACK streams that compete over the same radio resources as the TCP packets. In particular, they propose a new delayed ACK scheme in which the delay coefficient varies with the sequence number of the TCP packet. Simulations are used to show that the ACK thinning allows to increase TCP throughput substantially more than previous improvement methods.

Edge-based: R. de Oliveira et al. propose in [59, 60] a solution that only involves the end nodes in the congestion control performed by TCP. In other words, no specific cooperation from intermediate nodes is needed. Rather, a TCP sender monitors the network state and uses this input to react correctly. Using this approach, different monitoring techniques may be used to infer the internal network state. The authors propose to evaluate the reliability of using RTT variation monitoring as a

congestion indication. Further, the use of fuzzy logic theory for assisting the TCP error detection mechanism in such networks is investigated. An elementary fuzzy logic engine is presented as an intelligent technique for discriminating packet loss due to congestion from packet loss by wireless induced errors. The architecture of the proposed fuzzy-based error detection mechanism is also introduced and discussed. The full approach, for inferring the internal state of the network, relies on RTT measurements only. Hence, this is an end-to-end scheme which requires only the end nodes' cooperation.

Explicit Notification with enhanced Inter-layer Communication and control (ENIC): Sun and Man propose an end-to-end reliable data transport scheme in [61]. The goal is to solve the issues of TCP performance degradation and packet losses due to bit errors and route failures in mobile ad hoc networks. The scheme is referred to as ENIC mechanism. The authors introduce the new design methodology - enhanced inter-layer control mechanism to improve the reliability of data transport. Comprehensive effects of interactions among MAC, routing and TCP are considered. The scheme is claimed to provide a true end-to-end TCP maintenance and recovery method during the route failure.

Fractional Window Increment (FeW): Analyzing TCP's operation over IEEE 802.11 multi-hop ad hoc networks also involves a cross-layer study. In [62], the authors investigate the effect of congestion and MAC contention on the interaction between TCP and on-demand ad hoc routing protocol in the 802.11 ad hoc networks. The study reveals several problems stemming from lack of coordination and sharing in such networks. It is observed that TCP induces the over-reaction of routing protocol and hurts the quality of the end-to-end connection. Therefore, one of the critical sources of lowering TCP throughput lies in the TCP window mechanism itself. To fix this problem, the authors propose a FeW scheme for TCP to prevent the over-reaction of the on-demand routing protocol by limiting TCP's aggressiveness. The proposed scheme is applicable to a wide range of transport protocols using the basic TCP mechanism, and the protocol behavior is analytically tractable. Simulation results demonstrate that the proposed scheme can dramatically improve the TCP performance and the network stability in a variety of IEEE 802.11 multi-hop networks. For example, in some chain-like topologies, the proposed scheme outperforms basic TCP by over 90%, and recent related variants of TCP (ADTCP and Link RED (LRED)) by over 70%.

Fixed-Retransmission Timeout (RTO): A simple heuristic is proposed in [63], called Fixed RTO, to distinguish between route loss and network congestion and thereby improve the performance of the routing algorithms. The TCP sender is modified, employing a heuristic to distinguish between route failures and congestion without relying on feedback from other network nodes. When timeouts occur consecutively, i.e. the missing ACK is not received before the second RTO expires, this is taken to be evidence of a route loss. The unacknowledged packet is retransmitted again but the RTO is not doubled a second time. The RTO remains fixed until the route is re-established and the retransmitted packet is acknowledged.

Slow Congestion Avoidance (SCA): A number of recent research studies have explored ways to improve TCP throughput in mobile ad hoc networks by improving its interaction with the IEEE 802.11 MAC layer. In particular, the hidden terminal effects caused by interference can impact

TCP performance dramatically and have been dealt with in the past by restricting the maximum sending window size. In [64], the authors have developed a TCP variant, which instead, adjusts the sending rate increase to achieve competitive goodput for TCP connections, named SCA. Extensive simulations indicate that a slower sending rate increase during the congestion avoidance phase of TCP, leads to improved performance for TCP Reno, while eliminating the negative effects inherent in restricting the maximum sending window size. The work discusses the applicability of the TCP oriented solution to the hidden terminal effect, includes a performance comparison against existing solutions and discusses its performance merits under various mobility conditions.

TCP with Adaptive Pacing (TCP-AP): In [65], the authors introduce a novel congestion control algorithm for TCP over multi-hop IEEE 802.11 wireless networks implementing rate-based scheduling of transmissions within the TCP congestion window. It is shown how a TCP sender can adapt its transmission rate close to the optimum using an estimate of the current 4-hop propagation delay and the coefficient of variation of recently measured round-trip times. The novel TCP variant is denoted as TCP-AP. Opposed to previous proposals for improving TCP over multi-hop IEEE 802.11 networks, TCP-AP retains the end-to-end semantics of TCP and neither relies on modifications on the routing or the link layer nor requires cross-layer information from intermediate nodes along the path. A comprehensive simulation study using ns-2 shows that TCP-AP achieves up to 84% more goodput than TCP New-Reno, provides excellent fairness in almost all scenarios, and is highly responsive to changing traffic conditions.

TCP Detection of Out-of-Order and Response (TCP-DOOR) [66] is a new way to make TCP adapt to frequent route changes without relying on feedback from the network. It is based on TCP detecting out-of-order delivery events and inferring route changes from these events. Normal TCP performs poorly in ad hoc networks because of frequent route changes. In TCP-DOOR, the sender can distinguish route changes from network congestion by detecting out-of-order delivery, thereafter improve the performance of normal TCP by not invoking unnecessary congestion control. The simulation results showed that TCP-DOOR can improve the TCP throughput significantly, 50% on average. The approach does not rely on the feedback from lower layers or from the network. The feedback mechanism can be difficult to implement and expensive to deploy. The approach is purely end-to-end, where only the endpoints participate in the procedure to determine the network state. Another advantage is that this approach is also applicable in an environment having both ad hoc and fixed network. Assuming that sender S and receiver R are in two fixed networks interconnected by an ad-hoc network. The TCP connection between S and R faces the same problem of frequent route changes. As a pure end-to-end approach, TCP-DOOR will work well in this environment, but feedback-based approach will not. The obvious tradeoff is that a feedback-based approach is more accurate because the information is directly from the network. So the conclusion is, for improving TCP over ad hoc networks, the feedback-based approach should be used if available, otherwise, the approach can work on any environment and still deliver a significant improvement.

TCP-FIT[11] [67] is an improved TCP congestion control algorithm for wireless networks, and its

[11] The protocol name TCP-FIT is not an abbreviation.

performance is compared with existing state-of-the-art congestion control algorithms as well as an application layer Parallel TCP scheme. In TCP-FIT, N virtual TCP sessions are utilized in a single TCP connection. The algorithm uses both packet loss and queuing delay as inputs to the congestion window control algorithm. The congestion window of each sessions is controlled in an AIMD manner based on packet loss information, whereas the number N of virtual sessions is adjusted dynamically based on delay. Experimental results demonstrate significant performance improvements under various channel conditions. Compared with application layer Parallel TCP, the proposed algorithm has the additional advantage of not requiring changes to the application layer software.

TCP/Restricted Congestion Window Enlargement (TCP/RCWE): The authors of [68] propose the TCP/RCWE, an enhancement to improve TCP's performance in mobile multi-hop wireless ad-hoc networks. The enhancement is based on enabling TCP to distinguish between causes of packet loss, e.g., high bit error rate and congestion, and adapt TCP to the current network state. TCP adapts its behavior to the network by considering the RTT of the packets. In contrast to other studies on TCP, the authors not only consider the effects of node mobility and packet size on TCP's performance, but also the effects of medium disturbances, resulting in high bit error rates. The proposed enhancement improves TCP by adapting its behavior to the ad-hoc network environment. All simulations were performed with two different radio propagation models, and the TCP/RCWE is shown to be able to adapt the transmission behavior of the sender to the current network state.

TCP Veno [69] is a simple and effective end-to-end congestion control mechanism for dealing with random packet loss, which can be prominent in wireless access networks. Veno monitors the network congestion level and uses that information to decide whether packet losses are likely to be due to congestion or random bit errors. Specifically: (1) it refines the multiplicative decrease algorithm of TCP Reno by adjusting the slow-start threshold according to the perceived network congestion level rather than a fixed drop factor and (2) it refines the linear increase algorithm so that the connection can stay longer in an operating region in which the network bandwidth is fully utilized. Some results show that in typical wireless access networks with 1% random packet loss rate, throughput improvement of up to 80% can be demonstrated.

TCP Westwood & Westwood+: TCP Westwood [70], proposed in 2001, is a sender-side modification of the TCP congestion window algorithm that is intended to improve upon the performance of TCP Reno and TCP New-Reno (*WestwoodNR*) in wired as well as wireless networks. The improvement is most significant in wireless networks with lossy links, since TCP Westwood relies on end-to-end bandwidth estimation to discriminate the cause of packet loss (congestion or wireless channel effect) which is a major problem in TCP Reno. TCP Westwood does not require inspection and/or interception of TCP packets at intermediate (proxy) nodes. Rather, it fully complies with the end-to-end TCP design principle. TCP Westwood relies on end-to-end bandwidth estimation to discriminate the cause of packet loss (congestion or wireless channel effect). The estimate is then used to compute congestion window and slow start threshold after a congestion episode, that is, after three duplicate acknowledgments or after a timeout. The rationale of this strategy is simple:

in contrast with TCP Reno, which ”blindly” halves the congestion window after three duplicate ACKs, TCP Westwood attempts to select a slow start threshold and a congestion window which are consistent with the effective bandwidth used at the time congestion is experienced. The mechanism is named ”faster recovery”. The proposed mechanism is particularly effective over wireless links where sporadic losses due to radio channel problems are often misinterpreted as a symptom of congestion by current TCP schemes and thus lead to an unnecessary window reduction.

TCP Westwood+ [71] is an evolution of TCP Westwood. The main novelty of Westwood+ is the algorithm used to estimate the available bandwidth end-to-end, since it was discovered that the Westwood bandwidth estimation algorithm did not work well in the presence of reverse traffic, due to ACK compression.

4.3.2 TCP incompatible solutions

Preferential ACK retransmission: In [72], the authors describe a new technique for improving TCP performance in an ad hoc network that uses a table-driven type of routing protocol paying attention to short-duration link failure. In addition, the work evaluates the case in which the effect of the collision of a data packet and an ACK packet is suppressed by Delayed ACK and resending the ACK packet preferentially. The authors show through simulations that the combination of these improvements can increase the TCP throughput about 20%.

TCP-Dynamic Adaptive Acknowledgment (TCP-DAA): Multi-hop wireless networks based on the IEEE 802.11 MAC protocol are promising for ad hoc networks in small scale today. The 802.11 protocol minimizes the well-known hidden node problem, but does not eliminate it completely. Consequently, the end-to-end bandwidth utilization may be quite poor if the involved protocols do not interact smoothly. In particular, the TCP protocol does not manage to obtain efficient bandwidth utilization because its congestion control mechanism is not tailored to such a complex environment. The main problems with TCP in such networks are the excessive amount of both spurious retransmissions and contention between data and ACK packets for the transmission medium.

In [73], the authors propose a dynamic adaptive strategy for minimizing the number of ACK packets in transit and mitigating spurious retransmissions, named TCP-DAA. Using this strategy, the receiver adjusts itself to the wireless channel condition by delaying more ACK packets when the channel is in good condition and less otherwise. The technique not only improves bandwidth utilization, but also reduces power consumption by retransmitting much less than a regular TCP does. Extensive simulation evaluations show that the scheme provides very good enhancements in a variety of scenarios.

4.4 Changes affecting relaying nodes

Here, solutions that would represent changes to most of the network nodes if implemented, are presented.

Alleviate self-contention: In [74], the authors focus on self-contention – contention between packets

of the same transport layer connection along the path from source to destination. They observe that self-contention plays an important role in degrading the TCP performance in multi-hop wireless networks and that the use of the popular IEEE 802.11 MAC protocol exacerbates self-contention.

The authors propose and study two MAC-layer approaches to alleviate self-contention. The first approach, called Quick Exchange (QE), is designed with the intent of reducing the effects of inter-flow self-contention (e.g. between packets of the same connection traveling in opposite directions). The design of the second mechanism, called Fast-Forward (FF) is geared towards decreasing intra-flow self-contention (e.g. between packets of the same connection traveling in the same direction).

The proposed schemes are simulated and studied, and the authors observe that quick-exchange consistently improves network aggregate goodput. In contrast to the expectations, FF causes sporadic and often negative effects on goodput for TCP connections. Upon investigation they find that while the MAC is, in some respect, operating more efficiently, interactions with TCP's congestion control mechanism cause the goodput to degrade. Various effects that cause the respective behaviors with QE and FF are analyzed in detail.

Ad hoc TCP (ATCP) [75] is a TCP adaptation for MANETs where TCP in addition to receiving information about route failures, takes into account high BER. Based on the information provided by ECN or ICMP "Destination Unreachable" packets, ATCP, a layer between IP and TCP, will put TCP in the proper state. The three states are *persist*, *congestion control*, and *retransmit*.

Atra: Anantharaman et al. [76] investigate the impact of the mobility of nodes in an ad-hoc network on TCP's performance. The authors identify the key factors that contribute to TCP's performance degradation as TCP losses, MAC link failure detection latency, Link failure notification latency, and Route computation time. The authors show that the above factors contribute both in absolute terms and in terms of their impact on TCP's behavior. The article presents a proposal for a framework called Atra consisting of three easily implementable mechanisms at the medium access and routing layers that alleviates the impact of mobility on TCP's performance.

Cross-layer Congestion Control (C^3TCP): Kliazovich and Granelli present in [77] the problem of performance degradation of transport layer protocols due to the congestion of Wireless LANs (WLANs). Following the analysis of available solutions to this problem, C^3TCP is presented. The solution is able to obtain higher performance by gathering capacity information such as bandwidth and delay at the link layer. The method requires the introduction of an additional module within the protocol stack of the mobile node, able to adjust the outgoing data stream based on capacity measurements. Moreover, the authors present a proposal to support an optional field in the existing IEEE 802.11 protocol, in order to support the presented congestion control solution as well as many other similar approaches. The achieved results underline a good agreement with design considerations and high utilization of the available resources.

Congestion-Aware Routing (CAR): TCP sessions in ad hoc networks compete with each other for bandwidth. The use of shortest path routing can result in multiple TCP sessions being channeled via a few congested areas or hotspots. As a consequence, most of these multiple TCP sessions interfere

with each other and hence, experience significant performance degradations. Spatially separating the TCP sessions such that they inflict much lower interference effects on each other may provide gains in performance.

In [78], the authors first investigate the possibilities of achieving such gains by considering a centralized, ideal, and unrealistic congestion-aware routing approach. They find that spatial separation benefits are possible with the considered approach, and can especially help long (in terms of hop count) TCP connections. The solution is named Centralized Congestion-Aware Routing (CCAR). Further they consider the implementation of a distributed routing protocol named Distributed Congestion-Aware Routing (DCAR) to achieve the spatial separation benefits.

Due to practicalities, such as the need for the exchange of congestion state, the existence of stale congestion information and the creation of sub-optimal paths, the benefits due to spatial separation are considerably undermined. Both macroscopic simulations and microscopic studies of specific constructed examples are performed to understand the reasons and quantify the various effects with both the centralized and the distributed approaches. The studies suggest that achieving performance gains by spatially separating TCP sessions may be extremely difficult if not impossible in ad hoc networks.

COntention-based PAth Selection (COPAS): In [79], the authors address the capture problem of TCP, which is a result of the interplay between the MAC layer and TCP backoff policies. This interplay causes nodes to unfairly capture the wireless shared medium, hence preventing neighboring nodes to access the channel. The capture problem of exponential backoff-based MAC protocols (e.g., IEEE 802.11 and Floor Acquisition Multiple Access (FAMA)) has been shown to have a negative influence on the TCP performance over MANETs. This has again been shown to have major negative effects on TCP performance, comparable to the impact of mobility.

A novel algorithm is proposed, called COPAS, which incorporates two mechanisms to enhance TCP performance by avoiding capture conditions. First, it uses disjoint forward (sender to receiver for TCP data) and reverse (receiver to sender for TCP ACKs) paths in order to minimize the conflicts of TCP data and ACK packets. Second, COPAS employs a dynamic contention balancing scheme where it continuously monitors and changes forward and reverse paths according to the level of MAC layer contention, hence minimizing the likelihood of capture.

Through extensive simulation, COPAS is shown to improve TCP throughput by up to 90% while keeping the routing overhead low. Contention-balancing takes into consideration the number of MAC layer backoffs the nodes have experienced recently. COPAS can be deployed on top of any on-demand routing protocol, such as DSR and AODV.

Cross-layer Information Awareness (IA) techniques for TCP are presented in [80]. The authors propose to make routing protocols aware of lost data packets and ACKs and help reduce TCP timeouts for mobility-induced losses. Toward this end, they present two mechanisms: Explicit Packet Loss Notification (EPLN) and Best Effort ACK Delivery (BEAD). EPLN seeks to notify TCP senders about lost data packets. For lost ACKs, BEAD attempts to retransmit ACKs at either intermediate

nodes or TCP receivers. Upon route failures, ACKs are dropped silently. Therefore, TCP has to wait for timeouts. EPLN and BEAD reduce TCP timeouts for mobility-induced losses by exploiting cross-layer information awareness. With EPLN, intermediate nodes seek to notify TCP senders about lost packets so that TCP can start retransmission earlier. With BEAD, intermediate nodes or TCP receivers retransmit ACKs for lost ACKs in a best-effort way. Both mechanisms extensively use cached routes, without initiating route discoveries at any intermediate node. The two feedback mechanisms are applicable to any routing protocol, as they address general problems that occur at the network layer.

Data and ACK combined: Since a radio channel is shared among terminals in an ad hoc network, packet collisions are frequent. When transmitting packets using TCP, data and ACK packets are transmitted in opposite directions on the same radio channel. Therefore, frequent collisions are unavoidable, and this seriously degrades throughput. To reduce the likelihood of packet collisions when an intermediate node transmits both data and ACK packets, these two types of packet can be combined and transmitted at the same time to increase the efficiency of radio channel utilization.

In [81], the authors propose a new technique to improve TCP performance by combining data and ACK packets. The proposed technique is claimed to be applicable to generic ad hoc networks, although it is based on a Time Division Multiple Access (TDMA) MAC protocol. By means of a simulation using networks with various topologies, the authors claim that throughput can be improved by up to 60% by applying the proposed technique.

Explicit Link Failure Notification (ELFN): Holland et al. show in [82] that the legacy TCP performs poorly under mobility, and propose ELFN as a solution to this problem. ELFN is similar to TCP Feedback (TCP-F), but with ELFN the TCP is more active in the pause period, probing the network to see if the route has been restored.

Link RED (LRED)/Adaptive Pacing: In [83], the authors study TCP performance over multi-hop wireless networks that use the IEEE 802.11 protocol as the access method. Concluding through analysis and simulations that TCP is poor at exploiting spatial reuse, the authors propose two techniques, LRED and Adaptive Pacing. These techniques yield improvement in TCP throughput by 5 % to 30% in various simulated topologies. The authors also validate some simulation results through real hardware experiments.

The LRED algorithm is motivated by the observation that TCP can potentially benefit from the built-in dropping mechanism of the 802.11 MAC. The main idea is to further tune up wireless link's drop probability, based on the perceived link drops. LRED is a simple mechanism that, by monitoring a single parameter, *the average number of retries in the packet transmissions at the link-layer*, accomplishes three goals: a) It helps to improve TCP throughput, b) It provides TCP an early sign of network overload, and c) It helps to improve inter-flow fairness.

The goal of Adaptive Pacing is to improve spatial channel reuse, by distributing traffic among intermediate nodes in a more balanced way, while enhancing the coordination of forwarding nodes along the data path. The design is aimed at the IEEE 802.11 MAC, where a node is allowed to further

back-off an additional packet transmission time when necessary, in addition to its current deferral period (i.e. the random backoff, plus one packet transmission time). The extra backoff interval helps in reducing contention drops caused by exposed receivers, and extends the range of the link-layer coordination from one hop to two hops, along the packet forwarding path.

The LRED and Adaptive Pacing algorithms works together as follows: Adaptive pacing is enabled by LRED. When a node finds its average number of retries to be less than min_th, it calculates its backoff time as usual. When the average number of retries goes beyond min_th, Adaptive Pacing is enabled and the backoff period is increased by an interval equal to the transmission time of the previous data packet.

Loss Tolerant TCP (LT-TCP): TCP performance suffers substantially when packet error rates increase beyond a value of about 1% - 5%. In [84], the authors propose LT-TCP, an end-end mechanism to improve the TCP performance over networks comprised of lossy wireless link. The scheme separates the congestion indications from the wireless packet erasures by exploiting ECN. To overcome packet erasures, a dynamic and adaptive Forward Error Correction (FEC) scheme that includes adaptation of the MSS for TCP is employed. Redundancy is added in the form of proactive FEC which tunes itself to the measured error rate. The residual packet errors are handled by an enhanced retransmission scheme using reactive FEC repair packets to complement proactive FEC and SACK retransmission. Dynamically changing the MSS tailors the number of segments in the window for optimal performance. The scheme is built on top of TCP-SACK and depends on SACK and timeouts as a last resort.

Multipath-TCP: Lim et al. [85] investigate the TCP performance over a multipath routing protocol. Multipath routing can improve the path availability in mobile environment. Thus, it has a great potential to improve TCP performance in ad hoc networks under mobility. Previous research on multipath routing have mostly used UDP traffic for performance evaluation. When TCP is used, the authors find that most times, using multiple paths simultaneously may actually degrade TCP performance. This is partly due to frequent out-of-order packet delivery via different paths. They then test another multipath routing strategy called backup path routing. Under the backup path routing scheme, TCP is able to gain improvements when challenged with mobility. Further, related issues of backup path routing which can affect TCP performance are studied.

Non-work-conserving scheduling: Sometimes it would be desirable that ad hoc nodes can communicate with servers in wired networks to upload or download data in scenarios where wireless ad hoc networks are deployed. In these cases TCP connections will span both wireless ad hoc and wired domains. However, TCP often faces severe unfairness in this type of connection scenario, which forces some TCP flows to completely stop transferring any data despite all links being in good states.

In [86], the authors propose a simple scheduling scheme, which helps competing TCP connections to achieve fairness without much throughput loss. The algorithm is a simple non-work-conserving scheduling algorithm to work with the IEEE 802.11 MAC protocol, replacing the normal First In, First Out (FIFO) work-conserving scheduling scheme in ad hoc networks. Simulation results show

that our scheme successfully eliminates the extreme unfairness existing in several scenarios.

Neighborhood RED (NRED): The TCP unfairness in ad hoc wireless networks stems from the nature of the shared wireless medium and location dependency. Viewing a node and its interfering nodes as a ”neighborhood”, the aggregate of local queues at these nodes represents the distributed queue for this neighborhood. However, this queue is not a FIFO queue. Flows sharing the queue have different, dynamically changing priorities determined by the topology and traffic patterns. Thus, they get different feedback in terms of packet loss rate and packet delay when congestion occurs. In wired networks, the RED scheme was found to improve TCP fairness.

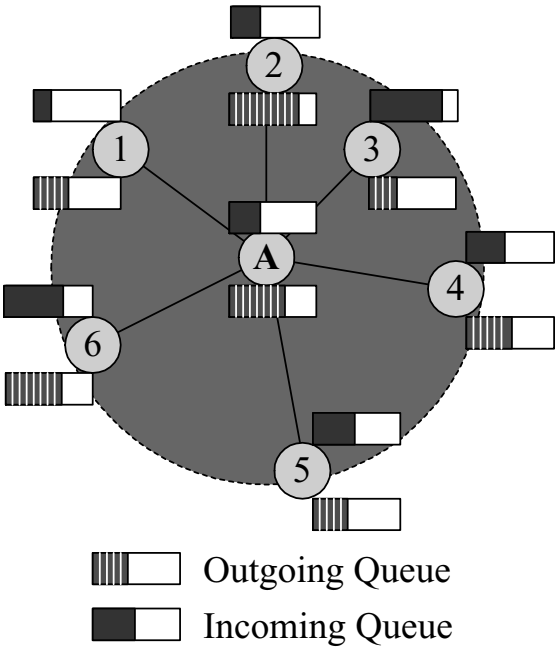

Figure 4.3 Simplified representation of the distributed neighborhood queue, from [87].

In [87], the authors show that the RED scheme does not work when running on individual queues in wireless nodes. Next, they propose a *NRED* scheme (Figure 4.3), which extends the RED concept to the distributed neighborhood queue. By detecting early congestion and dropping packets proportionally to a flow's channel bandwidth usage, the NRED scheme is able to improve TCP fairness. These two measurements can derive inputs needed for NRED implementation: 1) When a packet in any outgoing queue is transmitted, node A will detect the medium as busy. 2) If a packet is received to any incoming queue, node A can also learn this through the Clear To Send (CTS) packet (the IEEE 802.11 MAC layer is assumed).

Simulation studies confirm that the NRED scheme can improve TCP unfairness substantially in ad hoc networks. Moreover, the NRED scheme acts at the network level, without MAC protocol modifications. This considerably simplifies its deployment.

Optimum Packet scheduling for Each Traffic flow (OPET): In wireless multi-hop ad hoc networks, nodes need to contend for the shared wireless channel with their neighbors, which could result

in congestions and greatly decrease the end-to-end throughput due to severe packet loss. Several papers have indicated that the IEEE 802.11 fails to achieve the optimum schedule for this kind of contentions.

In [88, 89], the authors present a framework of multi-hop packet scheduling to achieve maximum throughput for traffic flows in the shared channel environment. The key idea is based on the observation that in the IEEE 802.11 MAC protocol the maximum throughput for chain topology is $\frac{1}{4}$ of the channel bandwidth and its optimum packet scheduling is to allow simultaneous transmissions at nodes which are four hops away. The proposed fully distributed scheme generalizes this optimum scheduling to any traffic flow that may encounter intra-flow contentions and inter-flow contentions. The objective of the scheme is to achieve OPET, and hence greatly increase end-to-end throughput and decrease end-to-end delay of multi-hop flows. By alleviating the intra-flow contention and inter-flow contention problems, the scheme, aptly named OPET greatly reduces the resource wasted by those dropped packets at forwarding nodes and thus could significantly improve the end-to-end performance. The solution addresses the MAC-layer.

Extensive simulations indicate that the scheme could perform well and achieve high throughput at light to heavy traffic load, while the performance of the original IEEE 802.11 MAC protocol greatly degrades when the traffic load becomes heavy. Moreover, the scheme also achieves much better and more stable performance in terms of delay, fairness and scalability with low and stable control overhead.

Preemptive routing: While not directly motivated by TCP's challenges in MANETs, preemptive routing deserves commenting, since it addresses an important challenge for TCP in MANETs. Goff et al. [90] propose adding proactive route selection and maintenance to on-demand ad-hoc routing algorithms. The authors propose algorithms that initiate proactive path switches when the quality of a path in use becomes suspect. When a path is likely to be broken, a warning is sent to the source indicating the likelihood of a disconnection. The source can then initiate path discovery early, potentially avoiding the disconnection altogether. A path is considered likely to break when the received packet power becomes close to the minimum detectable power (other approaches are possible). This proactivity avoids using a path that is about to fail and eliminates the associated costs of detecting the failure and recovering from it, significantly improving the performance of the network.

Another work that falls into the same category of preemptive routing is [91] by Larsen et al. It may be considered the proactive equivalent of [90]. The contribution of [91] is the proposal to divide the transmission area of a node into a safe zone and an unsafe zone (Figure 4.4), and distribute this information to all neighbors. A node (A) can select routes preferring neighbors in the safe zone, based on the knowledge of whether its neighbors are in the safe or unsafe zone. The nodes in the buffer zone are considered to be at a higher risk of disappearing during the next HELLO timeout period. If another node (B) exists that is both in the source node's safe zone and also has the node C in its own safe zone, it is considered safer to route packets via node B to node C. In other words, from (A)'s perspective, node (B) is a safe node and node (C) is an unsafe node.

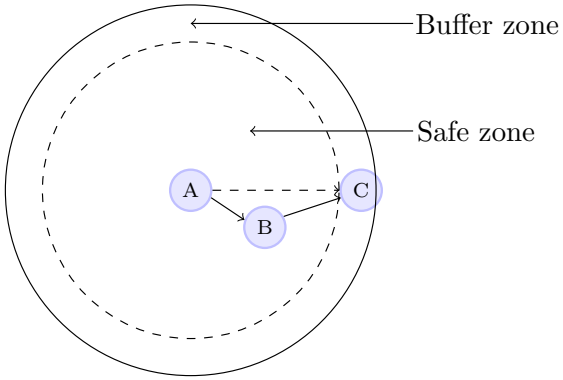

Figure 4.4 Transmission area zones.

Rate Based end-to-end Congestion Control (RBCC): TCP encounters a number of new challenges when applied in MANETs, such as wireless link error, medium contention, and frequent route failures. And very poor performance of TCP in MANET has been reported in many recent studies.

In [92], Zhai et al. focus on the problems resulting from the medium contention and propose a novel RBCC scheme. First they illustrate that, under the impact of medium contention, the window-based congestion control algorithm is unstable and hence may not be appropriate for MANETs, because the optimum congestion window size is very small and may be even less than one. I.e., the source should send less than one packet in one RTT. Based on the novel use of channel busyness ratio, which is shown to be an accurate sign of the network utilization and congestion status, a new rate control scheme has been proposed to efficiently and reliably support the transport service in MANET. In RBCC, a sub-layer consisting of a leaky bucket is added under TCP to control the sending rate based on the network layer feedback at the bottleneck node. Extensive simulations show that the scheme significantly outperforms traditional TCP in terms of channel utilization, delay, and fairness.

Signal strength based link management: In [93], the authors propose mechanisms that are based on signal strength measurements to alleviate packet losses due to mobility. The key ideas are (a) if the signal strength measurements indicate that a link failure is most likely due to a neighbor moving out of range, in reaction, facilitate the use of temporary higher transmission power to keep the link alive and, (b) if the signal strength measurements indicate that a link is likely to fail, initiate a route re-discovery proactively before the link actually fails. The authors make changes at the MAC and the routing layers to predict link failures and estimate if a link failure is due to mobility. In addition, they propose a simple mechanism at the MAC layer that can help alleviate false link failures, which occur due to congestion when the IEEE 802.11 MAC protocol is used.

The objective is to reduce the packet losses due to mobility in ad hoc networks and thereby improve the performance of TCP. A link management framework is proposed that helps in salvaging TCP packets in transit upon the incidence of link failure. The framework consists of three individual components. First, a temporary increase in the transmit power level is induced when a node moves out of range to temporarily re-establish the failed link. This would enable the TCP packets that are already in flight to traverse the link. The use of the IEEE 802.11 MAC protocol causes false link

failures due to congestion. The authors propose a mechanism that allows distinguishing between true link failures due to mobility and false link failures. This mechanism is based on the measurement of signal strength at the physical layer and is used to determine if a node is still within range. Power levels are then increased to temporarily reestablish a failed link only if it is determined to be mobility induced. A proactive scheme is included, in which weak links are identified based on these signal strength measurements and routes are proactively found prior to failure. This scheme in turn helps in switching to the new route even before the failure occurs and thus can stem packet losses. The proactive and reactive signal strength based schemes are unified with another simple MAC layer extension. With the extension, the MAC layer, upon perceiving false link failures, simply increases the number of Request To Send (RTS) attempts in order to salvage transit TCP packets. The authors recognize that additional mechanisms are necessary to correctly determine the levels of congestion of the network. These mechanisms can help to decide whether the reactive Link Management (LM) approach should be incorporated to salvage packets in transit since during heavy congestion and low mobility, temporary increases in transmission power can lead to some adverse effects.

TCP Buffering capability and Sequence information (TCP-BuS) [94] uses network layer feedback to detect route failure events and to take appropriate reaction to the event. The scheme introduces buffering capability in the mobile nodes. The scheme is based on the Associativity-Based Routing (ABR) protocol [95]. The node discovering a route failure is called the Pivoting Node (PN). Upon receiving notice of a disconnection, packets along the path from the source to the PN are buffered, and for these buffered packets, the retransmission timer is doubled. The lost packets along the path from the source to the PN are not retransmitted until the adjusted retransmission timer expires.

TCP Feedback (TCP-F) [96] is a TCP protocol based on feedback from the routing protocol to handle route failures in MANETs. A route failure notification is sent to the source by the routing agent on a node. The source freezes all variables for the TCP flow and puts the TCP flow on hold. The TCP flow is then resumed when a notification of the route reestablishment is given. To avoid blocking, the TCP protocol will timeout after a period without receiving a notification of reestablishment, invoking the normal congestion control algorithm.

TCP-Recomputation (TCP-RC) [97] is a proposal for TCP in MANETs where the TCP source is notified through an Explicit Route Failure Notification (ERFN) message when a route fails. This freezes the retransmission timers and pauses the congestion control. Upon the reconstruction of the route, cwnd and ssthresh are re-computed for the TCP connection. Thus, it can adjust the TCP transmission rate adaptively according to the current capacity of the TCP connection. Consequently, TCP-RC lowers the possibility of bursty traffic and avoids invoking congestion control during a situation of high network load.

4.5 Discussion

A significant number of works that address the problems faced by TCP in MANETs have been presented above. The solutions are not to be understood as products. In most cases the proposals

have merely been considered by the authors themselves through analytical and simulation studies. The simulation studies are often limited with regards to the types of scenarios in which the solutions are tested.

Many of the solutions undoubtedly contain elements that are employable in a mobile military network. However, it can be useful to study the needs of a mobile military network a bit closer. Employing a solution would require thorough testing, and it would be preferable if the solution is implemented as standard or configurable in most current OSs. Due to the large quantity of user equipment, the solution would need to be highly scalable, and allow for partial implementation.[12] It should be interoperable with other NATO nations' Networking and Information Infrastructure (NII). The solution would have to be implementable on many different types of equipment and must be compatible with the security architecture of the Norwegian Armed Forces (NAF).

Since the solution should allow partial implementation, it would be beneficial to know whether the solution is compatible with existing TCP implementations. If the solution is compatible as is, then it could be phased into the existing NII in smaller steps. Solutions incompatible with existing TCP will require an arbitrator mechanism to determine if the new incompatible TCP version can be used, if only some of the network nodes support the implementation.

Another way to solve a situation with a partially introduced TCP-for-MANET solution, is to "hide" the MANET behind a gateway. Thus, any traffic entering or exiting the MANET can be controlled at the border. TCP flows can be terminated from both sides at this GW. A new challenge that could emerge from such a solution is problems handling the flow control between the two networks. In addition, multi-homing of the MANET could force the GWs to exchange a lot of state information. In the short term, it may be difficult to implement a GW solution with extra TCP-for-MANET support within the existing security architecture, as the encryption equipment software handles the required GW functionality by itself, and this software has to be controlled by the security authorities.

A main question when considering implementing a solution is the impact of the solution with regards to the required changes to current equipment. One can consider several different degrees of change. The first is a case where no changes are needed at all, and where the solution can be enabled by simple configuration. Another case is where only key equipment, such as GWs, needs to be changed. However, it seems to be difficult to introduce a modified TCP solution without changing at least one of the end-points. The most common change impact seen in the proposed solutions is a need to alter the TCP protocol at the end-points. A more intrusive change is the requirement of changes even to the TCP flow relay nodes. The final, and most intrusive type of change, requires the entire network to implement certain behavior.

In Table 4.1, the presented solutions are evaluated against several criteria:

TCP compatible Is the solution compatible with current common implementations of TCP?
GW-oriented Is the solution oriented towards a GW improving the performance of TCP in the

[12] Partial implementation is desired in order to phase in the solution gradually, and avoid the need for all equipment to be changed at the same time.

MANET?

Change impact What changes must be implemented to enable the solution?

N No changes.

E Changes only to one or both of the end-nodes.

R Changes to the relay-nodes as well as the endpoints.[13]

As the table shows, most of the solutions are compatible with TCP implementations in current systems. This does not mean that the solutions will give performance gains if used with these non-MANET optimized TCP implementations, but rather that it is possible for two end-points, where one is a vanilla TCP implementation, to communicate.

Very few of the solutions are GW-oriented, but in the near future, such solutions may be the easiest to implement. They operate on the border of the MANET and may be configured to support the desired behavior of the MANET.

In intrusiveness, all the solutions require some change to the network nodes, except DSProxy in its most basic form. However, the DSProxy solution, as well as the other GW solutions, do require a changed addressing scheme, where the TCP flow is terminated at the GW instead of at the end-point in the MANET. Of the other solutions, many only require changes to the end-points, while some even need the relay nodes to implement changes. It seems that the GW-oriented solutions may offer the least change impact, even though they may require some change to the end-points in the MANET.

Most of the solutions address the incorrect behavior of the congestion control algorithm when facing losses due to other causes than network overload.

ECN has not been explicitly presented as a solution for improving TCP's performance in MANETs. However, it is employed in multiple solutions to explicitly signal congestion, where packet loss due to other causes than congestion is implicitly identified through the absence of a ECN. In such a case, the congestion control algorithm should not be triggered.

It is important to be aware that most TCP changes only affect the traffic sent from the node, and not the received traffic. Both ends must implement the change to ensure the proper behavior of other congestion control methods.

End-point nodes inside the MANET may be connected via a wired Ethernet cable to a radio or a router. This can pose a challenge for those solutions that depend on information flow between TCP and other layers, for instance communication with the routing protocol so TCP's parameters can be frozen in case of rerouting delay.

Communication in military networks often requires encryption to secure classified information. The current communication security architecture in the Norwegian NII is based on separating red enclaves (plain text subnets) with secured information and a black network core (cipher text) using encryption

[13] Solutions that require ECN signaling from relay nodes are classified as requiring changes to relay nodes, since the ECN mechanism is not universally available.

Table 4.1 Evaluation of TCP solutions for improving TCP performance in MANET.

Name	TCP compatible	GW-oriented	Change impact
ADHOCTCP	√		E
ADTCP	√		E
Alleviate self-contention			R
ATCP	√		R
C^3TCP	√		R
Cross-layer IA			R
Data and ACK combined			R
DSProxy	√	√	N/E
Dynamic congestion window limit	√		E
Dynamically delayed ACK	√		E
Edge-based	√		E
ELFN	√		E
ENIC	√		E
FeW	√		E
Fixed-RTO	√		E
LT-TCP			R
Preferential ACK retransmission			E
RBCC	√		R
SCA	√		E
Split-TCP		√	E
TCP-AP	√		E
TCP-BuS			R
TCP-DAA			E
TCP-DOOR	√		E
TCP-F	√		E
TCP-FIT	√		E
TCP-GAP	√	√	E
TCP-RC	√		R
TCP/RCWE	√		E
TCP Veno	√		E
TCP Westwood & Westwood+	√		E

devices[14]. Encrypted tunnels are established between encryption devices. This represents a problem for many of the proposed solutions, since very little information is allowed through the encryption device (between the red and black networks), and since the tunnels are point-to-point containing

[14]Many of the problems that the use of encryption devices pose in the network infrastructure are described in [98].

IP-packets with obscured payload. Thus, no relaying nodes in the black network may know whether the packet contains TCP-information. While the security architecture may change in the distant future, it is currently an architecture that needs to be taken into account, and it is interesting to consider whether the solutions are compatible with networks connected using encrypted tunnels.

This report has focused on solutions aimed at MANETs. A heterogeneous MANET may consist of a lot of different links, even sat-com links. TCP over sat-com is another field entirely, but may entail solutions with elements that could be employed in MANETs. A master's thesis [99] with work originating from FFI investigates the behavior of the TCP protocol over hybrid satellite networks. The thesis provides a thorough description of TCP, satellite environments, and related challenges in military networks containing both satellite and radio links. With an added delay of around 550 ms for geostationary satellites and high bit error rates, TCP performance may suffer severely, depending on the variant of TCP used. The thesis shows that due to fairness problems, the TCP variants available to Windows 7 perform poorly in lossy networks in competition with CUBIC and Hybla. Especially the latter outperforms the other variants in such environments. The result of competing on the same bottleneck is unfairness at the cost of Windows 7 TCP variants. Solutions are proposed, including tuning of the TCP variants, avoiding mixed-OS/TCP environments, or using a Performance Enhancing Proxy (PEP) called *PEPsal* at the sender side. However, PEPsal breaks the end-to-end principle of TCP, and depends on a plaintext TCP header – causing challenges in encrypted military networks.

5 Conclusions

This report has presented the functionality of TCP, in addition to its development for wired networks as the congestion window mechanism was invented and refined. The challenges that TCP faces in MANETs have been explained from a OSI network stack perspective. After this, a large number of different proposals to make TCP work better in MANETs were presented and discussed.

The discussion of the findings showed the complexity of the solutions and the problem of choosing one single solution to improve the TCP performance, and more research is definitely needed before any solution could be deployed as part of an operational network.

With the findings of this report taken into consideration, it is legitimate to ask whether TCP is the only choice for reliable communication in MANETs. There are efforts focusing on UDP-based communications that may handle this communication well enough, employing Automatic Repeat-reQuest (ARQ) at the MAC-layer and handling reliability at the application layer. In such a case, the congestion control offered by TCP must be handled with other mechanisms, through admission control and local Quality of Service (QoS). Another direction is within the research of Information Centric Networking (ICN), where congestion avoidance is also addressed, e.g. [100].

References

[1] H. Zimmermann, *OSI reference model—The ISO model of architecture for open systems interconnection*. Norwood, MA, USA: Artech House, Inc., 1988, ch. 1, pp. 2–9.

[2] W. R. Stevens, *TCP/IP illustrated (vol. 1): the protocols*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[3] W. R. Stevens and G. R. Wright, *TCP/IP illustrated (vol. 2): the implementation*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[4] W. R. Stevens, *TCP/IP illustrated (vol. 3): TCP for transactions, HTTP, NNTP, and the Unix domain protocols*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1996.

[5] P. D. Amer. (2008) TCP Variations: Tahoe, Reno, New Reno, Vegas, Sack. Computer & Information Sciences, University of Delaware. [Online]. Available: http://www.cis.udel.edu/~amer/856/TCP-Variations-Amer-for-Sethi-Simulation-08f.ppt

[6] N. Spring, D. Wetherall, and D. Ely, *RFC 3540: Robust Explicit Congestion Notification (ECN) Signaling with Nonces*, IETF Experimental, June 2003.

[7] K. Ramakrishnan, S. Floyd, and D. Black, *RFC 3168: The Addition of Explicit Congestion Notification (ECN) to IP*, IETF Internet Standard, September 2001.

[8] J. Postel, *RFC 793: Transmission Control Protocol*, IETF Internet Standard, Sep. 1981.

[9] V. Jacobson, R. Braden, and D. Borman, *RFC 1323: TCP Extensions for High Performance*, IETF Internet Standard, May 1992.

[10] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, *RFC 2018: TCP Selective Acknowledgment Options*, IETF Internet Standard, October 1996.

[11] V. G. Cerf and R. E. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, pp. 637–648, 1974.

[12] R. S. Tomlinson, "Selecting sequence numbers," *SIGOPS Oper. Syst. Rev.*, vol. 9, pp. 11–23, January 1975. [Online]. Available: http://doi.acm.org/10.1145/563905.810894

[13] Internet Engineering Task Force, R. Braden (ed.), *RFC 1122: Requirements for Internet Hosts - - Communication Layers*, IETF Internet Standard, October 1989.

[14] S. Deering and R. Hinden, *RFC 2460: Internet Protocol, Version 6 (IPv6) Specification*, IETF Internet Standard, December 1998.

[15] X. Xiao, A. Hannan, V. Paxson, and E. Crabbe, *RFC 2873: TCP Processing of the IPv4 Precedence Field*, IETF Internet Standard, June 2000.

[16] M. Allman, W. Paxson, and E. Blanton, *RFC 5681: TCP Congestion Control*, IETF Standards Track, September 2009.

[17] V. Paxson, M. Allman, J. Chu, and M. Sargent, *RFC 6298: Computing TCP's Retransmission Timer*, IETF Internet Standard, June 2011.

[18] M. Duke, R. Braden, W. Eddy, and E. Blanton, *RFC 4614: A Roadmap for Transmission Control Protocol (TCP) Specification Documents*, IETF Internet Standard, September 2006.

[19] J. Nagle, *RFC 896: Congestion Control in IP/TCP Internetworks*, IETF Internet Standard, January 1984.

[20] P. Karn and C. Partridge, "Improving round-trip time estimates in reliable transport protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 17, pp. 2–7, August 1987. [Online]. Available: http://doi.acm.org/10.1145/55483.55484

[21] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *SIGCOMM Comput. Commun. Rev.*, vol. 26, pp. 5–21, July 1996. [Online]. Available: http://doi.acm.org/10.1145/235160.235162

[22] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Comput. Commun. Rev.*, vol. 18, pp. 314–329, August 1988. [Online]. Available: http://doi.acm.org/10.1145/52325.52356

[23] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 4, pp. 397 –413, aug 1993.

[24] S. Floyd, "TCP and explicit congestion notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, pp. 8–23, October 1994. [Online]. Available: http://doi.acm.org/10.1145/205511.205512

[25] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: new techniques for congestion detection and avoidance," *SIGCOMM Comput. Commun. Rev.*, vol. 24, pp. 24–35, October 1994. [Online]. Available: http://doi.acm.org/10.1145/190809.190317

[26] Wikipedia.org. (2011, July) TCP Vegas. [Online]. Available: http://en.wikipedia.org/wiki/TCP_Vegas

[27] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," *SIGCOMM Comput. Commun. Rev.*, vol. 26, pp. 270–280, August 1996. [Online]. Available: http://doi.acm.org/10.1145/248157.248180

[28] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, *RFC 6582: The NewReno Modification to TCP's Fast Recovery Algorithm*, IETF Internet Standard, April 2012.

[29] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 4, march 2004, pp. 2514 – 2524 vol.4.

[30] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 64–74, July 2008. [Online]. Available: http://doi.acm.org/10.1145/1400097.1400105

[31] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-speed and Long Distance Networks," Microsoft Research, Tech. Rep., 2005.

[32] A. Medina, M. Allman, and S. Floyd, "Measuring the evolution of transport protocols in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 35, pp. 37–52, April 2005. [Online]. Available: http://doi.acm.org/10.1145/1064413.1064418

[33] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP Congestion Avoidance Algorithm Identification," in *Proceedings of IEEE ICDCS*, 2011. [Online]. Available: http://cse.unl.edu/~xu/paper/peng_ICDCS_2011.pdf

[34] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specification," IEEE standard 802.11, June 1999.

[35] ——, "Wireless LAN medium access control (MAC) and physical layer (PHY) specification," IEEE standard 802.11-2007, June 2007.

[36] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?" *Communications Magazine, IEEE*, vol. 39, no. 6, pp. 130–137, Jun 2001.

[37] S. M. Mirhosseini and F. Torgheh, *ADHOCTCP: Improving TCP Performance in Ad Hoc Networks*. InTech, 2011, ch. 7, pp. 121–138, isbn: 978-953-307-402-3.

[38] S. Papanastasiou, M. Ould-khaoua, and L. M. Mackenzie, "On the evaluation of TCP in manets," in *In Proceedings of the International Workshop on Wireless Ad Hoc Networks*, 2005.

[39] C. Perkins and E. Belding-Royer, "Ad-hoc On-Demand Distance Vector Routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999, pp. 90–100, New Orleans, LA.

[40] U. Ibom, "Tcp performance over manet," in *Information Networking, 2008. ICOIN 2008. International Conference on*, jan. 2008, pp. 1 –5.

[41] K. Pentikousis, "TCP in wired-cum-wireless environments," *Communications Surveys Tutorials, IEEE*, vol. 3, no. 4, pp. 2 –14, quarter 2000.

[42] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multi-hop Networks," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, ser. WMCSA '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 41–. [Online]. Available: http://portal.acm.org/citation.cfm?id=520551.837518

[43] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, vol. 9, no. 11, pp. 40:40–40:54, Nov. 2011. [Online]. Available: http://doi.acm.org/10.1145/2063166.2071893

[44] K. Nichols and V. Jacobson, “Controlling queue delay,” *Queue*, vol. 10, no. 5, pp. 20:20–20:34, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2208917.2209336

[45] V. Kawadia and P. R. Kumar, “A cautionary perspective on cross-layer design,” *Wireless Communications, IEEE*, vol. 12, no. 1, pp. 3–11, 2005. [Online]. Available: http://dx.doi.org/10.1109/MWC.2005.1404568

[46] M. Voorhaen and C. Blondia, “Analyzing the Impact of Neighbor Sensing on the Performance of the OLSR protocol,” in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006 4th International Symposium on*, April 2006, pp. 1–6.

[47] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, *RFC 3135: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*, IETF Informational, June 2001.

[48] H. Balakrishnan, V. N. Padmanabhan, G. Fairhurst, and M. Sooriyabandara, *RFC 3449: TCP Performance Implications of Network Path Asymmetry*, IETF Best Current Practice, December 2002.

[49] F. Wang and Y. Zhang, “A Survey on TCP over Mobile Ad-Hoc Networks,” in *Ad Hoc and Sensor Networks*, Y. Xiao and Y. Pan, Eds. Nova Science Publishers, 2005, pp. 267–281.

[50] A. Al Hanbali, E. Altman, and P. Nain, “A Survey of TCP over Mobile Ad Hoc Networks,” INRIA, Research Report RR-5182, 05 2004. [Online]. Available: http://hal.inria.fr/inria-00071406/en/

[51] C. Lochert, B. Scheuermann, and M. Mauve, “A survey on congestion control for mobile ad hoc networks: Research Articles,” *Wirel. Commun. Mob. Comput.*, vol. 7, pp. 655–676, June 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1255143.1255152

[52] E. Skjervold, T. Hafsoe, F. Johnsen, and K. Lund, “Delay and disruption tolerant Web services for heterogeneous networks,” in *Military Communications Conference, 2009. MILCOM 2009. IEEE*, 2009, pp. 1 –8.

[53] “ACP 142(A), P_MUL - A PROTOCOL FOR RELIABLE MULTICAST IN BANDWIDTH CONSTRAINED AND DELAYED ACKNOWLEDGEMENT (EMCON) ENVIRONMENTS,” The Combined Communications-Electronics Board (CCEB), Tech. Rep., October 2008.

[54] S. Kopparty, S. Krishnamurthy, M. Faloutsos, and S. Tripathi, “Split TCP for mobile ad hoc networks,” in *Global Telecommunications Conference, 2002. GLOBECOM ’02. IEEE*, vol. 1, 2002, pp. 138 – 142 vol.1.

[55] S. M. ElRakabawy, A. Klemm, and C. Lindemann, “Tcp with gateway adaptive pacing for multihop wireless networks with internet connectivity,” *Comput. Netw.*, vol. 52, no. 1, pp. 180–198, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2007.09.016

[56] Z. Fu, B. Greenstein, X. Meng, and S. Lu, "Design and implementation of a tcp-friendly transport protocol for ad hoc wireless networks," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, nov. 2002, pp. 216 – 225.

[57] K. Chen, Y. Xue, and K. Nahrstedt, "On setting TCP's congestion window limit in mobile ad hoc networks," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 2, may 2003, pp. 1080 – 1084 vol.2.

[58] E. Altman and T. Jiménez, "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks." in *PWC'03*, 2003, pp. 237–250.

[59] R. de Oliveira, T. Braun, and M. HeissenbÃ$\frac{1}{4}$ttel., "An Edge-based Approach for Improving TCP in Wireless Mobile Ad Hoc Networks." in *In DADS '03: Proceedings of the Conference on Design, Analysis and Simulation of Distributed Systems, Orlando, USA*, March 2003.

[60] R. de Oliveira and T. Braun, "A delay-based approach using fuzzy logic to improve TCP error detection in ad hoc networks," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 3, march 2004, pp. 1666 – 1671 Vol.3.

[61] D. Sun and H. Man, "Enic - an improved reliable transport scheme for mobile ad hoc networks," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 5, 2001, pp. 2852 –2856 vol.5.

[62] K. Nahm, A. Helmy, and C.-C. Jay Kuo, "TCP over multihop 802.11 networks: issues and performance enhancement," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '05. New York, NY, USA: ACM, 2005, pp. 277–287. [Online]. Available: http://doi.acm.org/10.1145/1062689.1062725

[63] T. D. Dyer and R. V. Boppana, "A comparison of tcp performance over three routing protocols for mobile ad hoc networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, ser. MobiHoc '01. New York, NY, USA: ACM, 2001, pp. 56–66. [Online]. Available: http://doi.acm.org/10.1145/501422.501425

[64] S. Papanastasiou and M. Ould-Khaoua, "TCP congestion window evolution and spatial reuse in MANETs: Research Articles," *Wirel. Commun. Mob. Comput.*, vol. 4, pp. 669–682, September 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1072160.1072167

[65] S. M. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with adaptive pacing for multihop wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '05. New York, NY, USA: ACM, 2005, pp. 288–299. [Online]. Available: http://doi.acm.org/10.1145/1062689.1062726

[66] F. Wang and Y. Zhang, "Improving tcp performance over mobile ad-hoc networks with out-of-order detection and response," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, ser. MobiHoc '02. New York, NY, USA: ACM, 2002, pp. 217–225. [Online]. Available: http://doi.acm.org/10.1145/513800.513827

[67] J. Wang, J. Wen, J. Zhang, and Y. Han, "TCP-FIT: A novel TCP congestion control algorithm for wireless networks," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, dec. 2010, pp. 2065 –2069.

[68] M. Gunes and D. Vlahovic, "The performance of the TCP/RCWE enhancement for ad-hoc networks," in *Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on*, 2002, pp. 43 – 48.

[69] C. P. Fu and S. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 2, pp. 216 – 228, feb 2003.

[70] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, ser. MobiCom '01. New York, NY, USA: ACM, 2001, pp. 287–297. [Online]. Available: http://doi.acm.org/10.1145/381677.381704

[71] L. A. Grieco and S. Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 25–38, April 2004. [Online]. Available: http://doi.acm.org/10.1145/997150.997155

[72] M. Sugano and M. Murata, "Performance improvement of tcp on a wireless ad hoc network," in *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, vol. 4, april 2003, pp. 2276 – 2280 vol.4.

[73] R. de Oliveira and T. Braun, "A dynamic adaptive acknowledgment strategy for TCP over multihop wireless networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, march 2005, pp. 1863 – 1874 vol. 3.

[74] D. Berger, Z. Ye, P. Sinha, S. Krishnamurthy, M. Faloutsos, and S. Tripathi, "Tcp-friendly medium access control for ad-hoc wireless networks: alleviating self-contention," in *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, oct. 2004, pp. 214 – 223.

[75] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 19, no. 7, pp. 1300 –1315, Jul. 2001.

[76] V. Anantharaman, S.-J. Park, K. Sundaresan, and R. Sivakumar, "Tcp performance over mobile ad hoc networks: a quantitative study: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 4, pp. 203–222, March 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1072597.1072604

[77] D. Kliazovich and F. Granelli, "Cross-layer congestion control in ad hoc wireless networks," *Ad Hoc Networks*, pp. 687–708, 2006.

[78] Z. Ye, S. Krishnamurthy, and S. Tripathi, "Use of congestion-aware routing to spatially separate tcp connections in wireless ad hoc networks," in *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, oct. 2004, pp. 389 – 397.

[79] C. Cordeiro, S. Das, and D. Agrawal, "COPAS: dynamic contention-balancing to enhance the performance of TCP over multi-hop wireless networks," in *Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference on*, oct. 2002, pp. 382 – 387.

[80] X. Yu, "Improving tcp performance over mobile ad hoc networks by exploiting cross-layer information awareness," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, ser. MobiCom '04. New York, NY, USA: ACM, 2004, pp. 231–244. [Online]. Available: http://doi.acm.org/10.1145/1023720.1023743

[81] T. Yuki, T. Yamamoto, M. Sugano, M. Murata, H. Miyahara, and T. Hatauchi, "Improvement of TCP Throughput by Combination of Data and ACK Packets in Ad Hoc Networks," *IEICE Transactions on Communications*, vol. E87-B, no. 9, pp. 2493–2499, September 2004.

[82] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Wirel. Netw.*, vol. 8, pp. 275–288, March 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1013798127590

[83] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on tcp throughput and loss," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, march-3 april 2003, pp. 1744 – 1753 vol.3.

[84] O. Tickoo, V. Subramanian, S. Kalyanaraman, and K. K. Ramakrishnan, "Lt-tcp: End-to-end framework to improve tcp performance over networks with lossy channels." in *IWQoS*, ser. Lecture Notes in Computer Science, H. de Meer and N. T. Bhatti, Eds., vol. 3552. Springer, 2005, pp. 81–93. [Online]. Available: http://dblp.uni-trier.de/db/conf/iwqos/iwqos2005.html#TickooSKR05

[85] H. Lim, K. Xu, and M. Gerla, "Tcp performance over multipath routing in mobile ad hoc networks," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 2, may 2003, pp. 1064 – 1068 vol.2.

[86] L. Yang, W. K. Seah, and Q. Yin, "Improving fairness among TCP flows crossing wireless ad hoc and wired networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, ser. MobiHoc '03. New York, NY, USA: ACM, 2003, pp. 57–63. [Online]. Available: http://doi.acm.org/10.1145/778415.778423

[87] K. Xu, M. Gerla, L. Qi, and Y. Shu, "TCP Unfairness in ad hoc wireless networks and a neighborhood RED solution," *Wirel. Netw.*, vol. 11, pp. 383–399, July 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-005-1764-1

[88] H. Zhai, J. Wang, and Y. Fang, “Distributed Packet Scheduling for Multihop Flows in Ad Hoc Networks,” in *Proc. IEEE WCNC’04*, 2004, pp. 1081–1086.

[89] H. Zhai, X. Chen, and Y. Fang, “Alleviating intra-flow and inter-flow contentions for reliable service in mobile ad hoc networks,” in *Military Communications Conference, 2004. MILCOM 2004. IEEE*, vol. 3, oct.-3 nov. 2004, pp. 1640 – 1646 Vol. 3.

[90] T. Goff, N. B. Abu-ghazaleh, D. S. Phatak, and R. Kahvecioglu, “Preemptive Routing in Ad Hoc Networks,” *Proceedings ACM/IEEE MobiCom*, vol. 1, pp. 43–52, 2001.

[91] E. Larsen, L. Landmark, V. Pham, O. Kure, and P. E. Engelstad, “Routing with Transmission Buffer Zones in MANETs,” in *10th IEEE International Symposium on a ”World of Wireless, Mobile and Multimedia Networks”, WoWMoM*, June 2009.

[92] H. Zhai, X. Chen, and Y. Fang, “Rate-based transport control for mobile ad hoc networks,” in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 4, march 2005, pp. 2264 – 2269 Vol. 4.

[93] F. Klemm, Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, “Improving tcp performance in ad hoc networks using signal strength based link management,” *Ad Hoc Netw.*, vol. 3, pp. 175–191, March 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1640913.1640956

[94] D. Kim, C.-K. Toh, and Y. Choi, “TCP-BuS: Improving TCP Performance in Wireless Ad Hoc Networks,” in *Journal Of Communications And Networks*, 2001, pp. 1707–1713.

[95] C.-K. Toh, “Associativity-Based Routing For Ad-Hoc Mobile Networks,” in *Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems*. Kluwer Academic Publishers, March 1997, vol. 4, no. 2, pp. 103–139.

[96] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, “A Feedback Based Scheme for Improving TCP Performance in Ad-Hoc Wireless Networks,” in *Proceedings of the The 18th International Conference on Distributed Computing Systems*, ser. ICDCS ’98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 472–. [Online]. Available: http://portal.acm.org/citation.cfm?id=850926.851742

[97] J. Zhou, B. Shi, and L. Zou, “Improve TCP performance in ad hoc network by TCP-RC,” in *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on*, vol. 1, sept. 2003, pp. 216 – 220 Vol.1.

[98] G. Elmasry, M. Jain, R. Welsh, K. Jakubowski, and K. Whittaker, “Network management challenges for joint forces interoperability,” *Communications Magazine, IEEE*, vol. 49, no. 10, pp. 81 –89, oct. 2011.

[99] A. R. Urke, “Transport Layer challenges in hybrid military satellite networks,” Master’s thesis, University of Oslo, Department of Informatics, 2011.

[100] F. Bjurefors, P. Gunningberg, C. Rohner, and S. Tavakoli, “Congestion avoidance in a data-centric opportunistic network,” in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ser. ICN ’11. New York, NY, USA: ACM, 2011, pp. 32–37. [Online]. Available: http://doi.acm.org/10.1145/2018584.2018594

Abbreviations

ABR	Associativity-Based Routing
ACK	Acknowledgment
ADHOCTCP	Ad Hoc TCP
ADTCP	Ad Hoc TCP
AIAD	Additive Increase, Additive Decrease
AIMD	Additive Increase, Multiplicative Decrease
AODV	Ad hoc On-Demand Distance Vector Routing
API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
ARQ	Automatic Repeat-reQuest
ATCP	Ad hoc TCP
BDP	Bandwidth-Delay Product
BEAD	Best Effort ACK Delivery
BER	Bit Error Rate
BIC	Binary Increase Congestion control
BSD	Berkeley Software Distribution
C^{3}TCP	Cross-layer Congestion Control
CA	Collision Avoidance
CAR	Congestion-Aware Routing
CCAR	Centralized Congestion-Aware Routing
COPAS	COntention-based PAth Selection
COTS	Commercial Off-The-Shelf
CSMA	Carrier Sense Multiple Access
CTCP	Compound TCP
CTS	Clear To Send
CWL	Congestion Window Limit

CWR	Congestion Window Reduced
D-SACK	Duplicate Selective Acknowledgment
DAMA	Demand Assigned Multiple Access
DCAR	Distributed Congestion-Aware Routing
DSR	Dynamic Source Routing
ECE	ECN-Echo
ECN	Explicit Congestion Notification
ELFN	Explicit Link Failure Notification
EMCON	Emission Control
ENIC	Explicit Notification with enhanced Inter-layer Communication and control
EPLN	Explicit Packet Loss Notification
ERFN	Explicit Route Failure Notification
FAMA	Floor Acquisition Multiple Access
FEC	Forward Error Correction
FeW	Fractional Window Increment
FF	Fast-Forward
FIFO	First In, First Out
FIN	Finish
GW	Gateway
HPC	High Performance Computing
IA	Information Awareness
ICMP	Internet Control Message Protocol
ICN	Information Centric Networking
ICW	Initial Congestion Window
IDD	Inter-packet Delay Difference
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force

IP	Internet Protocol
LLN	Link Layer Notification
LM	Link Management
LRED	Link RED
LT-TCP	Loss Tolerant TCP
MAC	Medium Access Control
MANET	Mobile Ad hoc NETwork
MS	MicroSoft
MSL	Maximum Segment Lifetime
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NAF	Norwegian Armed Forces
NATO	North Atlantic Treaty Organization
NII	Networking and Information Infrastructure
NRED	Neighborhood RED
OPET	Optimum Packet scheduling for Each Traffic flow
OS	Operating System
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PEP	Performance Enhancing Proxy
PLR	Packet loss ratio
PMUL	A protocol for reliable multicast messaging in bandwidth constrained and delayed acknowledgement (EMCON) environments
PN	Pivoting Node
POR	Packet out-of-order delivery ratio
PSH	Push
QE	Quick Exchange

QoS	Quality of Service
RBCC	Rate Based end-to-end Congestion Control
RED	Random Early Detection
RFC	Request For Comments
RTHC	Round-Trip Hop-Count
RTO	Retransmission Timeout
RTS	Request To Send
RTT	Round Trip Time
SA	Situational Awareness
SACK	Selective Acknowledgment
SCA	Slow Congestion Avoidance
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
STT	Short Term Throughput
SYN	Synchronise
TCP	Transmission Control Protocol
TCP-AP	TCP with Adaptive Pacing
TCP-BuS	TCP Buffering capability and Sequence information
TCP-DAA	TCP-Dynamic Adaptive Acknowledgment
TCP-DOOR	TCP Detection of Out-of-Order and Response
TCP-F	TCP Feedback
TCP-GAP	TCP Gateway Adaptive Pacing
TCP-RC	TCP-Recomputation
TCP/RCWE	TCP/Restricted Congestion Window Enlargement
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UHF	Ultra High Frequency

VHF	Very High Frequency
VoIP	Voice over IP
WLAN	Wireless LAN
WWW	World Wide Web
YeAH-TCP	Yet Another High-speed TCP