Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

A Reliable Transport Protocol for Unreliable Networks

TIK Report 239

Simon Heimlicher, Rainer Baumann, Martin May, Bernhard Plattner

Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology, Switzerland
{heimlicher,baumann,may,plattner}@tik.ee.ethz.ch

8th February 2006

Abstract

Unstable networks such as wireless mobile ad hoc networks (MANETs) are radically different from traditional fixed networks like the Internet. The success of these emerging networks is determined by the performance of the communication layers and the transport layer. For the latter, many protocols have been proposed based on adaptations of TCP for such unstable networks. In this paper, we claim that the underlying principles of TCP are not suitable for unstable networks and propose a fundamentally different reliable transport protocol called *SAFT*. SAFT employs the store-and-forward principle to enable data transfer across networks with intermittent end-to-end connectivity. Using simulations, we show that SAFT realizes faster data transmission and uses network resources more efficiently than TCP. Furthermore, SAFT behaves more fairly when multiple connections compete for network resources in the same collision domain. We argue that our transport protocol results in several benefits including robustness, flexibility, and fairness.

1 Introduction

The fast growing segment of wireless networks has radically different properties from established fixed networks such as the Internet. Mobility of nodes and unreliability of links are the dominant properties of these emerging networks. As a result, routes in such networks are very unstable and hence, a fundamental design paradigm of the Internet is questioned: the *end-to-end principle* [1], [2]. This principle has led the development of the quasi-standard transport protocol for reliable transmission in packet-switched networks, the *Transmission Control Protocol (TCP)* [3].

Hop-by-hop packet forwarding is an alternative to end-to-end transport protocols. We will discuss in this paper that the hop-by-hop forwarding paradigm is of particular interest for unstable networks, such as mobile ad hoc networks (MANETs).

So far, most reliable transport protocols proposed for MANETs were modifications of TCP (see [4], [5], [6], [7], [8], and [9]). Therefore, they are all window-based, end-to-end protocols.

There are also a few protocols that replaced the window-based congestion and flow control mechanism of TCP with a rate-based approach, as for example [10].

However, only a fraction of the protocols currently under discussion propose hop-by-hop packet delivery. One protocol is window-based [11], another employs a rate-based mechanism [12].

Looking at the four mentioned properties, we illustrate the resulting design space for reliable transport protocols in the following table.

	Window-based	Rate-based
End-to-end	TCP, TCP-ELFN, ATCP, TCP-F, TCP-PR	ATP
Hop-by-hop	Split TCP	DTN, SAFT

In this paper, we propose the SAFT framework, a framework for rate-based, hop-by-hop transport protocols. SAFT stands for Store-And-Forward Transport. We show, that this approach is suitable for reliable transport, particularly in unstable networks. The SAFT approach is especially useful for delay-insensitive applications, such as multimedia messaging or file distribution.

Store-and-forward transport protocols transfer data in management units that comprise multiple packets. In this paper, we call these units *segments*. Segments are transferred through the network as follows. The source of a connection splits data into segments and transfers them to the next hop in direction of the destination. On every intermediate hop, incoming packets are stored until an entire segment has arrived. Then, the segment is again transferred to the next hop on the path, until it reaches the destination of the connection.

1.0.1 Why hop-by-hop?

The main advantage of a hop-by-hop protocol is that it handles local problems locally. This is especially useful for networks with mobile nodes, since node mobility induces route changes and route failures. This instability of the network leads to intermittent connectivity, hence, during a substantial fraction of time, there is no end-to-end route available between some or all pairs of nodes. In contrast, a hop-by-hop transport protocol is still able to use some of the active links to transfer data in direction of the destination.

Packets are cached at intermediate hops and resent if necessary. Compared to end-to-end protocols, packet losses are handled and solved locally and therefore, time and end-to-end bandwidth is saved. Also, the local problem handling reduces the probability of MAC layer collisions in other parts of the network. Furthermore, flow control between two communicating nodes is optimized according to the local situation, resulting in a better use of local communication resources and higher utilization of the total network bandwidth.

1.0.2 Why rate-based?

With per-hop flow control, a rate-based scheme has several properties that are useful for wireless networks. Today's wireless network technology, for instance, IEEE 802.11 [13], employs omnidirectional antennas. Hence, the collision domain is large and interference among neighbors is high. Hence, data transmission is expensive. With a window-based scheme for flow control, packet transmission is triggered by reception of an acknowledgement packet (ACK). Thus, the sender needs feedback from the receiver on a regular basis, and the system is *self-clocked*. A rate-based scheme is not self-clocking, but needs a clock at the sender to trigger transmissions. However, this allows the sender to continue transmitting for some time before it needs feedback from either the receiver or from other neighboring nodes.

Wherever possible, we use the experience gained from TCP in fixed networks and employ the well-established mechanisms in our hop-by-hop algorithm.

The rest of this paper is organized as follows. We next discuss related work. Then, we describe the protocol framework in Section 3. We present results of our simulation experiments in Section 4 and conclude the paper in Section 5.

2 Related Work

Extended analysis on the performance of TCP over mobile ad hoc networks ([4], [5], [6], and [14]) revealed that TCP is not suitable for unstable networks. TCP fails to respond appropriately to events that are frequent in unstable networks, such as topology changes, node disconnections, and link errors. Modifications of the primary TCP mechanisms have been proposed to address these problems.

In MANETs, packet losses mostly occur due to link errors and route failures rather than interface buffer overflows at intermediate nodes. However, the end points of a connection are unable to discern if packet losses are due to the above events or due to congestion. In [4] (TCP-ELFN), an Explicit Link Failure Notification (ELFN) technique is proposed, based on direct feedback about link and route failures from intermediate nodes to the sender. A similar approach is proposed in [5] (TCP-F). ATCP [6] tries to resolve this problem utilizing network layer feedback.

While the aforementioned approaches rely on cooperation and feedback control messages from lower layers, [7] employs a heuristic technique to distinguish route failures and congestion. When timeouts occur consequently due to unacknowledged data packets, the retransmission timeout (RTO) is not doubled. With this method, the authors tried to alleviate large timeouts at the sender that degrade TCP performance. Analogous approaches are proposed in [8] (TCP-DOOR), and [9] (TCP-PR) where out-of-order delivery of data packets at the destination or acknowledgement messages sent back to the sender are used as an indication of route failures.

While the TCP-variant protocols discussed above utilize window-based flow control, the Ad hoc Transport Protocol (ATP) [10] adopts a rate-based flow control mechanism. ATP uses information from lower layers for the detection, congestion avoidance and control, estimation of the transmission rate, and detection of path failures. The intermediate nodes attach congestion information to every ATP packet and the corresponding ATP receiver is responsible to send the accumulated information back to the sender in the next ACK packet.

With hop-by-hop protocols, all involved nodes need to run an instance of the protocol. Also, to enable hop-by-hop transfer, cross-layer communication with the routing layer is necessary. Furthermore, intermediate nodes have to process cross traffic not only at the network layer, but also at the transport layer.

The authors of [11] observed that long TCP connections suffer severely from route failures due to mobility. The Split TCP scheme separates long connections at intermediate nodes—proxies—into shorter segments. If a node is a proxy, it intercepts TCP packets, buffers them, and acknowledges them to the previous hop with a local acknowledgement (LACK) packet. It then forwards the buffered packets to the next hop. Upon receipt of a LACK, a proxy node will purge the packet from its buffer. To guarantee end-to-end reliability, the destination sends an acknowledgement back to the source.

An entirely different approach to the transport problem is taken by the Delay Tolerant Networking group [12]. In its extreme case, the authors consider the network as an interplanetary Internet. The Bundling protocol runs on top of established transport protocols like TCP and puts them to service to deliver a bunch of packets (called Bundle) to the destination. Due to the assumption that some of the underlying connections are episodic, intermediate hosts have to buffer all incoming bundles until they were successfully forwarded.

3 The SAFT Protocol Framework

In this section, we outline the design of our store-and-forward transport protocol (SAFT). After a high-level overview, we address the fundamental design ideas behind SAFT. Note that SAFT is implemented in a highly adaptable simulation framework. We use the term SAFT to refer to the protocols that are provided by this framework.

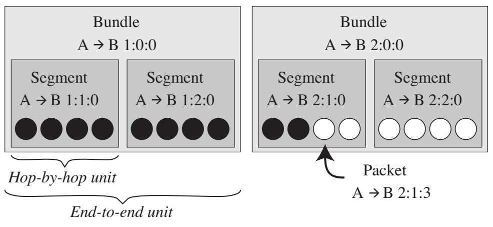


Figure 1: ***SAFT management units.*** *Source and destination use* bundles*, neighboring nodes use* segments *as management units*

3.1 Overview

From a high-level perspective, SAFT is a reliable transport layer implementation where the transport responsibility is transferred from the end systems to the intermediate nodes along the path from the source to the destination. For the transfer, the data is broken down into bundles and segments instead of packets, as illustrated in Figure 1. Each bundle consists of multiple segments and each segment contains multiple packets. Segments are transferred and acknowledged hop-by-hop, while the bundles are acknowledged end-to-end. For such hop-by-hop protocols, each hop requires information about the next hop towards the destination node. We assume that the underlying routing protocol provides this information and notifies the transport layer whenever a route to a requested destination is discovered or changes. Furthermore, compared to classical packet forwarding, SAFT requires additional processing power to process packets of cross connections at every intermediate hop. SAFT also requires buffer space at intermediate hosts. The amount of buffer space is limited by several mechanisms, as discussed in the next section.

3.2 Basic Protocol Design

From its beginning, SAFT was primarily designed for unstable networks. In particular, SAFT was built to overcome the limitations of traditional end-to-end transport protocols such as TCP. The SAFT protocol strives to (i) provide resilience to unstable network conditions; (ii) enable transmission over intermittent end-to-end routes; (iii) reduce end-to-end control traffic.

Before we discuss the design ideas of SAFT, we illustrate the fundamental operation principle of SAFT with the following example: Source node A transmits data to destination node B. For simplicity reasons we expect no packet losses, no cross traffic, and perfect connectivity.

- Source node A splits the data into bundles, and the bundles into segments
- Source A looks up the address of the next hop to the destination, H_1
- Source A transmits the segment to node H_1
- Intermediate hop H_1 sends a segment acknowledgement to the previous hop, A, when it has received the complete segment
- Every intermediate hop H_i forwards the segment to its next hop, H_{i+1}, in the same manner
- Once a complete bundle has arrived at the destination B, this node sends a bundle ACK to the source A.

We next discuss the three main duties of a reliable transport protocol: reliable delivery, flow control, and congestion control. SAFT handles these tasks independently.

3.2.1 Reliable Delivery

End-to-end reliability can only be ensured by the end points of the connection unless intermediate hops are assumed to be stable and reliable. While in the scenario of [12] intermediate hosts guarantee delivery, such an assumption would not be appropriate in mobile, unstable networks. In such networks, the mobility of the nodes is likely to cause network partitions. Further, nodes may lose connectivity completely or fail at any time. Therefore, the source retransmits a bundle that has not been acknowledged by the destination within an adjustable time-frame.

3.2.2 Flow Control

Hop-by-hop protocols have an advantage over end-to-end protocols since they precisely control the data transmission rate between two neighboring nodes on the data path. This ability is crucial in wireless networks where, due to the large collision domain, the bandwidth per square meter is limited and transmission is expensive. Since the performance of SAFT depends to a large extent on how fast a segment is transferred over a link, this process is highly optimized. Also, information about transmission rate and round-trip time (RTT) to neighboring nodes are shared among all connections using the same next hop, leading to a fair allocation of the available network resources. Note that up-to-date RTT information is as crucial as the transmission rate because it determines, when unacknowledged segments are retransmitted. Bad RTT estimates may lead to premature retransmissions when parts of the segment are still in the interface queue, and precious bandwidth is wasted.

To avoid overrunning of the receiver's buffer, the receiver sends a source quench message to the sender before it runs out of buffer space.

3.2.3 Congestion Control

Hop-by-hop protocols using caching techniques are prone to cause congestion because copies of packets may exist at multiple nodes. If acknowledgement packets are lost, data packets are retransmitted from several points within the network, leading to heavy network load and causing further packets to be lost and retransmitted.

SAFT prevents this vicious circle by its congestion control mechanism. This mechanism is based on a sliding-window algorithm similar the the one employed by TCP. The settings of two window sizes control the network load: (i) The *segment window* limits the number of segments that any host is allowed to send to a next hop without waiting for acknowledgements. Note that this is only a soft limit, i.e., if the next hop changes, the host sends again a full segment window to the new next hop. Also, any segment with a lower sequence number than the last sent segment may be retransmitted. (ii) The *bundle window* at the source controls the number of unacknowledged bundles. This enforces a hard limit on the total number of packets in the network per connection.

3.3 Resilience Features

As discussed in Section 2, packet loss, route changes, and route failures are responsible for the poor performance of TCP in unstable networks. In the remainder of the section, we discuss how SAFT provides resilience against these events.

3.3.1 Packet Loss

Since SAFT is a hop-by-hop protocol, it handles packet losses locally. That is, if a packet is lost on any intermediate link, the node at the receiving side will not acknowledge the corresponding segment and the last hop will retransmit the segment.

If a segment acknowledgement packet is lost and a segment is retransmitted even though the receiver has it in its cache already, the receiver immediately sends an ACK to the sender. In addition, every segment ACK contains the last four sequence numbers of the last received segments. Thus, in general, a segment can be acknowledged by four independent ACK packets. The same mechanism is employed for bundle ACKs. In addition, we learned from simulations that sending a second, redundant bundle ACK after a small delay further improves the protocol performance.

3.3.2 Route Changes

The above mechanisms may fail if parts or the whole route change. In order to avoid stale packets to congest the network, every data and ACK packet contains a so-called *final acknowledgement number*, i.e, the sequence number of the last segment received in sequence at the destination. Whenever a node receives a higher final acknowledgement number, it updates its status and flushes all segments with lower sequence numbers. This effectively controls the cache sizes at intermediate hops and provides another redundant acknowledgement channel from the destination back to the source. Furthermore, if route changes lead to situations where a node receives a stale packet from another node, it immediately stops the node by sending a final acknowledgement number update.

3.3.3 Route Failures

To remain independent from the routing layer, SAFT does not rely on the routing protocol providing accurate link state information. Rather, before a segment is transmitted, a small probe packet is sent to the next hop and only if the next hop replies, a segment transmission is started. While this function may duplicate mechanisms available in some routing protocols, it helps to prevent the sender interface queue from overflowing. Thus, if the interface queue is full, the probe is dropped instead of a data packet.

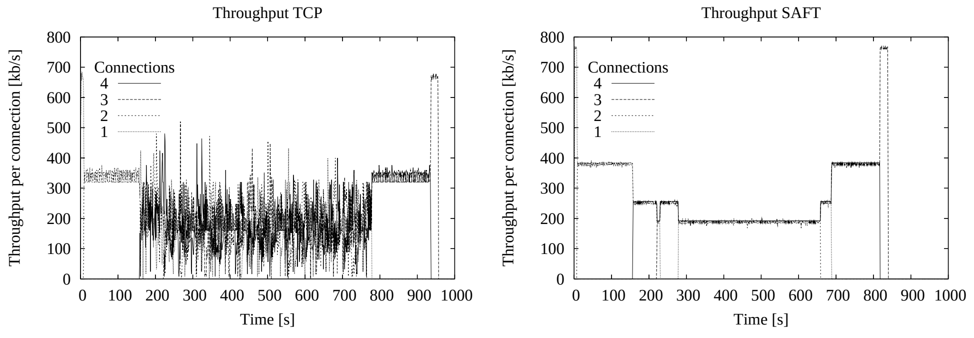
Current MANET routing protocols, such as AODV and DSR, strive to provide end-to-end routes. If a single link fails, the whole route is marked as invalid. Since hop-by-hop protocols do not depend on end-to-end routes, this functionality is counterproductive. Therefore, SAFT employs a route caching mechanism to continue to use routes as long as the next hop is up, even if the end-to-end route is marked invalid.

4 Evaluation

In this section, we present preliminary results from our simulation studies. Some properties are first examined in a static environment and later verified and validated in mobile scenarios. We also define the metrics we used throughout our studies and describe the chosen parameter settings of SAFT.

4.1 Simulation Environment

We use the network simulator "ns-2" for all experiments. The distributed coordination function (DCF) of IEEE 802.11 [15] is used at the MAC layer. The 802.11 DCF uses an RTS/CTS handshake for unicast transmissions to neighboring nodes for packets larger than a given threshold



(a) TCP

(b) SAFT

Figure 2: ***Single hop.*** *Throughput per connection. TCP vs. SAFT*

(set to 512 bytes). Broadcast data packets and the RTS/CTS control packets are sent using an un-slotted CSMA technique with collision avoidance (CSMA/CA). The radio model uses characteristics similar to a commercial radio interface, Lucent's "WaveLAN" card.

All nodes run the AODV [16] routing protocol, specifically the "AODV-UU" [17] implementation for ns-2. We compare SAFT with TCP NewReno [3], as implemented in the ns-2 "TCP/-FullTcp/Newreno" agent. We use an FTP application with a packet size of 1000 bytes.

4.1.1 Metrics

The goal of SAFT protocols is to provide a high level of connectivity to all pairs of nodes. Consequently, our main overall metric is *throughput per connection*. We use the same metric also to illustrate fairness among multiple connections.

4.1.2 SAFT Parameters

The SAFT framework offers a great number of knobs and dials. The most important parameters are the bundle and segment sizes. Learned from earlier studies, we set the bundle length to four segments and segment length to eight packets.

The simulation environment and the complete set of parameters is described and discussed in [18].

4.2 Static Scenarios

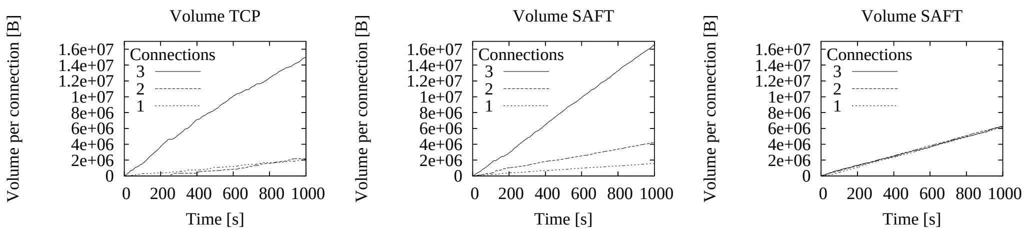
The static scenarios serve to demonstrate particular characteristics of SAFT in an observable environment.

4.2.1 Single-hop connection

We examine, how SAFT and TCP share the link resources among four single-hop connections. The source of each connection sends two messages of 10'000 packets, i.e., 10^7 bytes, to the destination at random start times $t_{1,2} = \text{uniform}(0, 500)$sec. The throughput of each connection for TCP is plotted in Figure 2(a), and for SAFT in Figure 2(b).

During the first seconds, connection 1 achieves a throughput of around 700kb for both SAFT and TCP. As soon as connection 2 starts sending, the throughput is cut down to half for both protocols. Apparently, TCP manages to share the link among two connections. However, as soon as more than two connections are active and compete for the link, the throughput of TCP varies heavily. In contrast, SAFT fairly shares the link bandwidth, independent of the number of connections. Note also, that the transmission time for all messages is about ten percent smaller with SAFT compared to TCP.

4.2.2 Multiple-hop connection



(a) TCP (b) SAFT (c) SAFT, bundle window $\rightarrow \infty$

Figure 3: ***Multiple hops.*** *Volume per connection. Connection 1 crosses 5 hops, connection 2 crosses 4 hops, and connection 3 crosses 3 hops.*

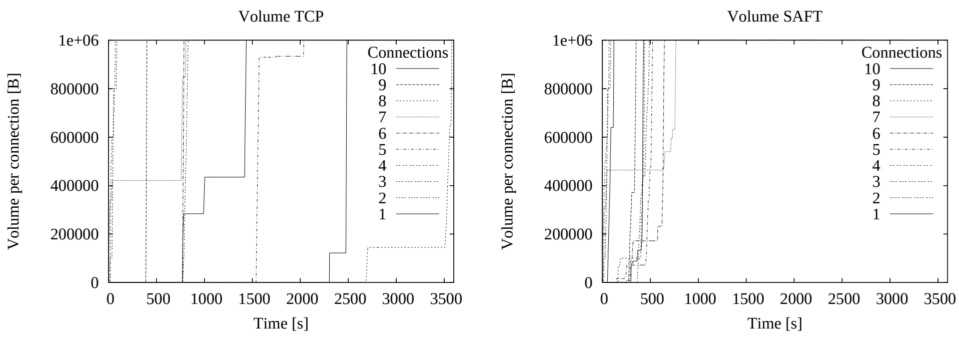
When several TCP connections share one link, TCP devotes more bandwidth to connections with smaller RTTs (cf. [11]). We examined this behavior of TCP and SAFT in the following scenario: Six nodes are placed in a row with a distance of 200m. Three connections are in place: Connection 1 between node 1 and 6, connection 2 between node 2 and 6 and connection 3 between node 5 and 1. In Figure 3(a), we plot how much data the destinations received per connection using TCP. While the three-hop connection 3 transfers about 15MB, connections 1 and 2 only obtain a share of 2MB.

With SAFT, as shown in Figure 3(b), the bandwidth of the links is shared more equally. The remaining unfairness is due to the fact that every hop increases the probability of link and route failures. [1] Hence, data and ACK packets are lost more frequently and congestion control limits the transmission rate at the sender. The congestion control mechanism of SAFT controls the number of unacknowledged packets in the network with the bundle window parameter. If this parameter is set to infinity, the fairness increases, as shown in Figure 3(c).

4.3 Mobile Scenario

We show an example scenario of a messaging application in a mobile ad hoc network. In an area of 1000m × 3000m, nodes move according to the Random-waypoint model [19]. Every node chooses a random point and a speed of 1–10m/sec. It then moves to this point and chooses the next waypoint, and so on. Each node in a set of ten randomly selected nodes connects to one other node that is not in this set. During the first 100 seconds, ten messages of 100kB each are sent to this node at uniformly distributed times. For network sizes of 10, 30 and 50 nodes, we have run 50 experiments with different seeds, resulting in different movement and connection patterns. We summarize the results in the table below.

[1]With AODV, every packet loss leads to a failure of the route between the end points of the link where the packet failure occurred.



(a) TCP (b) SAFT

Figure 4: *Random-waypoint model.* *Volume per connection*

Number of nodes	20	30	50
TCP Average [s]	1045.8	1212.8	443.5
SAFT Average [s]	364.0	372.1	317.2
TCP Std. dev.	1164.0	1549.4	411.6
SAFT Std. dev.	401.8	256.9	228.6
TCP Max. [s]	8602	4987	1970
SAFT Max. [s]	2091	1084	992

In Figure 4, we show the progress of the ten connections in one particular setting with 30 nodes. The plots show, how the volume of data received by the destinations of the connections increases over time. With TCP (cf. Figure 4(a)), it takes up to one hour until all messages have arrived at the destination nodes. With SAFT however, all connections finish the transmission within the first 15 minutes (cf. Figure 4(b).

5 Conclusion

In this paper, we presented an alternative to the dominant end-to-end communication paradigm for unstable networks. On the basis of the findings in related work, we developed a framework for a rate-based, hop-by-hop data forwarding algorithm, the **store-and-forward transport (SAFT)** protocol. We discussed the key design ideas and presented preliminary results of our simulation study.

We argue that SAFT leads to better utilization of networking resources in unstable networks. Further, SAFT introduces a “perfect” hop-by-hop fairness in presence of competing connections. With regard to the increased computing and memory efforts in the intermediate nodes, we believe that the behavior and performance of SAFT justifies the newly introduced overhead. In the future, we expect many ad hoc network applications that would greatly benefit from hop-by-hop transport protocols such as SAFT.

References

[1] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-To-End Arguments in System Design,” in *ACM Transactions on Computer Systems*, November 1984.

[2] D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols," in *Proceedings of the ACM Symposium on Communications Architectures and Protocols (SIGCOMM '88)*, August 1988, pp. 106–114.

[3] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 3782 (Proposed Standard), Apr. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3782.txt

[4] G. Holland and N. H. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," in *MobiCom '99*, August 1999.

[5] 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 International Conference on Distributed Computing Systems (ICDCS'98)*, May 1998.

[6] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad hoc Networks," *IEEE Journal on Selected Areas of Communication (JSAC)*, April 2001.

[7] M. Zhang, B. Karp, and S. Floyd, "RR-TCP: A Reordering-Robust TCP with DSACK," ICSI—International Computer Science Institute at Berkeley, CA, Tech. Rep. TR-02-006, July 2002.

[8] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response," in *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002)*, June 2002.

[9] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka, "TCP-PR: TCP for Persistent Packet Reordering," in *Proceedings of the IEEE 23rd International Conference on Distributed Computing Systems (ICDS 2003)*, May 2003, pp. 222–231.

[10] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, "ATP: A Reliable Transport Protocol for Ad-hoc Networks," in *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2003)*, June 2003.

[11] S. Kopparty, S. Krishnamurthy, M. Faloutsos, and S. Tripathi, "Split TCP for Mobile Ad Hoc Networks," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2002)*, November 2002.

[12] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-Tolerant Networking: An Approach to Interplanetary Internet," *IEEE Communications Magazine*, June 2003.

[13] M. S. Gast, *802.11 Wireless Networks—The Definitive Guide*, 1st ed. O'Reilly & Associates, Inc., Apr. 2002.

[14] T. Dyer and R. Boppana, "A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad-hoc Networks," in *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2001)*, October 2001.

[15] "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE standard 802.11–1997," 1997.

[16] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), July 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[17] “University of Uppsala AODV implementatioin—AODV-UU,” source code available: http://core.it.uu.se/AdHoc/ImplementationPortal.

[18] S. Heimlicher, R. Baumann, M. May, and B. Plattner, “SAFT—Store And Forward Transport in Mobile Ad-hoc Networks,” Communication Systems Group, ETH, Zurich, Tech. Rep., July 2005.

[19] C. Bettstetter, H. Hartenstein, and X. P’erez-Costa, “Stochastic Properties of the Random Waypoint Mobility Model,” in *ACM/Kluwer Wireless Networks: Special Issue on Modeling and Analysis of Mobile Networks*, September 2004.